HAL open science

Contributions to Decentralized and Privacy-Preserving Machine Learning

Aurélien Bellet

To cite this version:

Aurélien Bellet. Contributions to Decentralized and Privacy-Preserving Machine Learning. Machine Learning [cs.LG]. Université de Lille, 2021. tel-03542802v2

HAL Id: tel-03542802

https://theses.hal.science/tel-03542802v2

Submitted on 7 Feb 2022 (v2), last revised 22 Feb 2022 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Contributions to Decentralized and Privacy-Preserving Machine Learning

Université de Lille

Habilitation à Diriger des Recherches

Spécialité : Informatique

présentée par

Aurélien Bellet

Soutenue publiquement à Lille le 30/11/2021 après avis des rapporteurs,

Francis	**Bach**	Inria / École Normale Supérieure
Kamalika	**Chaudhuri**	University of California San Diego
Catuscia	**Palamidessi**	Inria

et devant le jury composé de :

Francis	**Bach**	Inria / École Normale Supérieure
Claude	**Castelluccia**	Inria / CNIL
Kamalika	**Chaudhuri**	University of California San Diego
Brendan	**McMahan**	Google Research
Catuscia	**Palamidessi**	Inria
Peter	**Richtárik**	King Abdullah University of Science and Technology
Adam	**Smith**	Boston University
Marc	**Tommasi**	Université de Lille

Abstract

This manuscript presents, in a unified way, some of my contributions to the topic of decentralized and privacy-preserving machine learning. Decentralized learning, also known as federated learning, aims to allow a set of participants with local datasets to collaboratively train machine learning models while keeping their data decentralized. A key challenge in this context is to design decentralized algorithms that (i) can efficiently solve a variety of learning tasks on highly heterogeneous local datasets, and (ii) provide rigorous privacy guarantees while minimizing the impact on the utility of the learned models. To tackle these challenges, I describe three sets of contributions. First, I present a decentralized approach to collaboratively learn a personalized model for each user. Second, I address the problem of decentralized estimation and learning with pairwise loss functions. In both cases, privacy-preserving versions of these algorithms are introduced under the strong model of local differential privacy. Finally, to reduce the cost in utility induced by local differential privacy, I propose two approaches to improve the privacy-utility trade-offs of decentralized learning through appropriate relaxations of the local model.

Keywords: decentralized learning; federated learning; differential privacy.

Résumé

Ce manuscrit présente, de manière unifiée, certaines de mes contributions sur le thème de l'apprentissage automatique décentralisé et respectueux de la vie privée. L'apprentissage décentralisé, également appelé apprentissage fédéré, permet à un ensemble de participants avec des données locales d'entraîner des modèles d'apprentissage de manière collaborative tout en gardant leurs données décentralisées. Un défi important dans ce contexte est de concevoir des algorithmes décentralisés qui (i) peuvent résoudre efficacement une variété de tâches d'apprentissage sur des données locales hétérogènes, et (ii) donnent des garanties de confidentialité rigoureuses tout en minimisant l'impact sur l'utilité du modèle appris. Pour relever ces défis, je décris trois ensembles de contributions. Premièrement, je présente une approche décentralisée pour apprendre, de manière collaborative, un modèle personnalisé pour chaque utilisateur. Deuxièmement, j'aborde le problème de l'estimation et de l'apprentissage décentralisés avec des fonctions de perte sur des paires. Dans les deux cas, des versions de ces algorithmes respectueuses de la vie privée sont introduites, en considérant le modèle local de la confidentialité différentielle. Enfin, pour réduire le coût en utilité induit par la confidentialité différentielle locale, je propose deux approches pour améliorer les compromis vie privée-utilité de l'apprentissage décentralisé via des relaxations appropriées du modèle local.

Mots clés : apprentissage décentralisé ; apprentissage fédéré ; confidentialité différentielle.

Acknowledgments

First of all, I would like to thank the members of the committee, especially the reviewers, for their time and consideration. It was a great pleasure and honor to present my work to leading experts on the various dimensions involved in my work, from both academia and industry. The Q&A session was intense and full of valuable feedback.

My deepest thanks go to Marc Tommasi. Beyond our fruitful scientific collaborations (some of which are presented in this thesis), I have learned a lot from Marc about how to be a researcher, from supervising PhD students to prioritizing things that actually matter. He has always encouraged me to pursue the directions he knew were important to me. As head of the Magnet team, he has also shielded me (and others) from a lot of administrative burden so I could focus more on my research.

I would like to thank all of my collaborators: a great part of the pleasure I get from doing research comes from these scientific and social interactions.

Finally, I would like to thank my family, in particular my parents and my brother for their continued interest and support, and my wife Marion, who always has my back.

Table of Contents

1 Introduction **9**

1.1 Key Concepts and Notations . . . 10

1.2 Overview of Featured Contributions . . . 13

1.3 Other Contributions . . . 15

2 Decentralized and Private Learning of Personalized Models **17**

2.1 Proposed Formulation . . . 19

2.2 Learning Personalized Models for Fixed Graph . . . 21

2.2.1 Decentralized Coordinate Descent Algorithm . . . 21

2.2.2 Communication-Efficient Decentralized Frank-Wolfe Algorithm . . . 24

2.3 Learning the Graph for Fixed Models . . . 26

2.4 Incorporating Differential Privacy Constraints . . . 29

2.4.1 Privacy Model . . . 29

2.4.2 Privacy-Preserving Decentralized Coordinate Descent . . . 29

2.4.3 Differential Privacy for the Full Alternating Procedure . . . 31

2.5 Experiments . . . 32

3 Decentralized and Private Learning with Pairwise Loss Functions **35**

3.1 Gossip Algorithms for Pairwise Estimation . . . 37

3.2 Gossip Algorithms for Pairwise Optimization . . . 40

3.2.1 Reminder on Centralized Dual Averaging . . . 41

3.2.2 Proposed Approach . . . 42

3.3 Locally Private Protocols for Pairwise Estimation . . . 44

3.3.1 Problem Setting . . . 44

3.3.2 Generic Locally Private Protocol from Quantization . . . 45

3.3.3 Locally Private Protocol for Area under the ROC Curve . . . 47

3.3.4 Beyond Local DP: Protocols from 2-Party Secure Computation . . . 51

3.4 Experiments . . . 52

4 Better Privacy-Utility Trade-offs for Decentralized Learning **57**
4.1 An Accurate, Scalable and Verifiable Protocol for Decentralized Differentially Private Averaging . . . 60
4.1.1 Problem Setting . . . 60
4.1.2 Proposed Protocol . . . 61
4.1.3 Differential Privacy Guarantees . . . 63
4.1.4 Correctness Against Malicious Users . . . 66
4.2 Privacy Amplification by Decentralization . . . 67
4.2.1 Network Differential Privacy . . . 68
4.2.2 Decentralized Computation Model . . . 69
4.2.3 Privacy Amplification for Walking on a Ring . . . 69
4.2.4 Privacy Amplification for Walking on a Complete Graph . . . 72
4.2.5 Experiments on Private Stochastic Gradient Descent . . . 76

5 Future Research **79**

Bibliography **85**

Appendix **101**
A Reminders on Differential Privacy . . . 101
B Reminders on Notions of Regularity and Curvature . . . 102

Chapter 1

Introduction

Machine Learning (ML) is at the heart of the innovative AI-based services and technology deployed on the Internet, in people's home and in public space. It is also routinely used in many data-oriented scientific fields such as medicine, neuroscience, biology, economics, marketing and the social sciences. State-of-the-art ML models, notably deep neural networks, must be trained on a large number of observations to be able to learn complex patterns that generalize well to unseen data. In the classic approach to ML, one assumes *centralized* access to the training data, be it on a single machine or in a tightly coupled system like a data center or "the cloud".

This stands in contrast to the fact that data is inherently *decentralized* in many use-cases of ML: medical data is collected by several hospitals, consumer data is collected by different companies, user data is generated by personal devices, etc. Collecting and storing data from multiple parties on a central server can incur high communication and infrastructure costs, and may not even be feasible in data-intensive applications.[1] Perhaps more crucially, centralized ML assumes the existence of a *trusted curator* who securely collects, stores and processes the raw data. As modern devices collect ever more sensitive personal data (e.g., browsing/purchase logs, geolocation data, speech utterances) and privacy scandals regularly make the news, many users of digital services are looking for more privacy-preserving solutions that do not require to hand over their data to the service provider. Situations where data owners may not be willing to share their raw data also arise when keeping control of data represents a competitive advantage (e.g., in business or research), and when data sharing is restricted by legal constraints (related to intellectual property or data privacy regulations).

Decentralized learning, also known as *federated learning*, has recently emerged as a way to address the limitations of centralized ML. This fast-growing research area aims to allow a set of participants with local datasets to collaboratively train machine learning models while keeping their data decentralized [**AB-Journal2**]. Decentralized learning

[1] For instance, a self-driving car is expected to generate several terabytes of data a day (Heinrich, 2017).

effectively moves most of the computation from the data center to where data is naturally located (e.g., on user devices). Like their centralized counterparts, decentralized learning algorithms are typically iterative procedures: participants update the current model in parallel using their local data, and these updates are then aggregated to form a new version of the model (McMahan et al., 2017). The aggregation step can be performed globally through communication with a central coordinator, or locally via peer-to-peer exchanges among subsets of users in the network (Lian et al., 2017).

Decentralized learning comes with its own set of challenges, which are distinct from those arising in traditional parallel/distributed ML designed to run in a cluster environment (see e.g., Bekkerman et al., 2011). I highlight here two main challenges (see the survey **[AB-Journal2]** for a more complete discussion). First and foremost, *local datasets exhibit large heterogeneity* as they reflect the usage and production patterns specific to each participant. In other words, they are not independent and identically distributed (*non-IID*). Intuitively, this makes decentralized learning harder because participants may largely disagree about how the model should be updated (Karimireddy et al., 2020). An important challenge is then to design decentralized algorithms that can solve a variety of ML tasks on highly heterogeneous local datasets, while operating under low communication budget and scaling gracefully with the number of participants.

A second challenge in decentralized learning is related to *protecting the privacy of participants* with respect to other parties. Avoiding to share raw data is a good starting point to design privacy-preserving solutions but is not sufficient to obtain robust privacy guarantees. In centralized ML, where a trusted curator only releases the trained model, information about individual training data points can be extracted from the model or its predictions (Shokri et al., 2017). This information can turn out to be very sensitive, as illustrated for instance by my work on reconstructing genotypes in private genomic databases from genetic risk score models **[AB-Conf6]**; **[AB-Journal3]**. Compared to the centralized setting, decentralized ML provides an additional attack surface as participants get to observe intermediate model updates instead of only the final model (Nasr et al., 2019). A key challenge is thus to design decentralized algorithms with rigorous privacy guarantees, minimizing the impact on the utility (accuracy) of the resulting models.

To summarize, it is crucial to design ML algorithms that can learn from heterogeneous decentralized datasets under rigorous privacy requirements, and to formally characterize the underlying trade-offs between utility, privacy, and efficiency. This manuscript presents some of my contributions towards achieving this goal.

1.1 Key Concepts and Notations

In this section, I introduce the main technical concepts and notations that will be used throughout the manuscript. Given an integer n, I will use $[\![n]\!]$ to denote the set $\{1, \ldots, n\}$.

Decentralized learning. Consider a set of n users (participants) involved in a decentralized learning process. Unless otherwise noted, users are assumed to behave honestly (i.e., they follow the protocol). Each user $u \in [\![n]\!]$ holds a local dataset $\mathcal{D}_u$ composed of $m_u = |\mathcal{D}_u|$ data points lying in a data domain $\mathcal{X}$ which is kept abstract for now. Each local dataset $\mathcal{D}_u$ should be thought of as being drawn from a local distribution specific to user u. Crucially, the joint dataset $\mathcal{D} = \cup_{u=1}^{n} \mathcal{D}_u$ of size $m = |\mathcal{D}| = \sum_{u=1}^{n} m_u$ will never be centralized: it will remain distributed across users.

I will focus on machine learning problems that can be formulated in the Empirical Risk Minimization (ERM) framework, which is motivated by statistical learning theory and encompasses many existing ML tasks (Shalev-Shwartz and Ben-David, 2014). Formally, consider a family of models with parameters $\theta \in \mathcal{C}$, representing for instance the weights of a linear classifier or deep neural network. Given a differentiable loss function $f : \mathcal{C} \times \mathcal{X} \to \mathbb{R}_+$ measuring how well a model $\theta \in \mathcal{C}$ fits a data point $x \in \mathcal{X}$, a standard ML objective is to solve the following optimization problem:

$$\arg\min_{\theta \in \mathcal{C}} \Big\{ F(\theta; \mathcal{D}) = \frac{1}{m} \sum_{x \in \mathcal{D}} f(\theta; x) \Big\}. \tag{1.1}$$

In other words, the goal is to find the parameters that minimize the average loss over the joint dataset $\mathcal{D}$. In the context of decentralized learning, the objective function $F(\theta)$ in (1.1) can be equivalently formulated as a weighted average of local objectives $F_1(\theta; \mathcal{D}_1), \ldots, F_n(\theta; \mathcal{D}_n)$ that each depend on the data of one user:

$$F(\theta; \mathcal{D}) = \sum_{u=1}^{n} \frac{m_u}{m} F_u(\theta; \mathcal{D}_u), \quad \text{where } F_u(\theta; \mathcal{D}_u) = \frac{1}{m_u} \sum_{x \in \mathcal{D}_u} f(\theta; x). \tag{1.2}$$

To collaboratively find an (approximate) solution to (1.1) in a decentralized fashion, users can perform local computations and communicate results by sending messages to other parties. Leveraging the separable structure in (1.2), the local computations of each user u will typically consist of gradient updates to the model with respect to his/her local objective F_u, while communication steps aim to aggregate local model parameters across users. Communication can rely on a *central coordinator* in a standard client-server architecture organized in a star topology. In this case, users send messages (i.e., model updates) to the coordinator, who aggregates them and sends the result (i.e., the new version of the model) back to the users (McMahan et al., 2017). Another approach is to consider a *fully decentralized* setting: users exchange information directly in arbitrary network topology represented as a graph $\mathcal{G} = ([\![n]\!], E)$ where nodes correspond to users and an edge $\{u, v\} \in E \subseteq [\![n]\!] \times [\![n]\!]$ indicates that users u and v can exchange messages. Fully decentralized algorithms are only able to perform local aggregations among neighboring users in $\mathcal{G}$, but generally scale better to the large number of users seen in "cross-device" applications [**AB-Journal2**]. Indeed, while the central coordinator may become a bottleneck as the number of users increases, the topology $\mathcal{G}$ used in fully decentralized algorithms can

remain sparse enough (i.e., with constant or logarithmic degree) such that all users need only to communicate with a small number of other users (Lian et al., 2017). Most of the contributions presented in this manuscript consider the fully decentralized setting.

Differential privacy. In this work, privacy will be measured by Differential Privacy (DP), a mathematical definition which comes with rigorous guarantees as well as a powerful algorithmic framework. First introduced by Dwork et al. (2006b), DP has become a gold standard metric of privacy in fundamental science but also in real-world deployments, as evidenced by its recent adoption by the US Census Bureau (Abowd, 2018) and some big tech companies (Erlingsson et al., 2014; Fanti et al., 2016; Ding et al., 2017). DP relies on a neighboring relation over pairs of datasets, which can be adapted to the context. The relation most commonly used in ML says that two datasets $\mathcal{D}$ and $\mathcal{D}'$ of same size are neighboring, denoted by $\mathcal{D} \sim \mathcal{D}'$, if they differ on a single data point.

Definition 1.1 (Differential privacy)**.** *Let $\mathcal{A}$ be a (randomized) algorithm taking a dataset as input. Given real parameters $\epsilon, \delta \geq 0$, we say that $\mathcal{A}$ is (ϵ, δ)-differentially private, or (ϵ, δ)-DP, if for all possible pairs of neighboring datasets $\mathcal{D} \sim \mathcal{D}'$ and for all $\mathcal{O} \subseteq \mathrm{range}(\mathcal{A})$, we have:*

$$\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{O}) \leq e^{\epsilon} \Pr(\mathcal{A}(\mathcal{D}') \in \mathcal{O}) + \delta. \tag{1.3}$$

Differential privacy requires that the distribution of the possible outputs of $\mathcal{A}$ is almost the same regardless of whether a particular data point was part of the input. This formalizes the intuitive requirement that the output of a private algorithm should not reveal too much information about individual data points. The requirement gets stronger as ϵ and δ approach 0, and the special case of $\delta = 0$ is known as *pure* ϵ-DP. DP possesses a number of desirable properties, among which *robustness to post-processing* (any function of an ϵ-DP algorithm remains ϵ-DP) and *composition* (the ability to keep track of privacy guarantees across multiple analyses). For instance, given K algorithms that each satisfy (ϵ, δ)-DP, releasing their combined outputs on the same data is $(K\epsilon, K\delta)$-DP.

Crucially, $\mathcal{A}$ must be randomized to satisfy DP. This gives rise to a classic *trade-off between privacy and utility*, where privacy is measured by (ϵ, δ) and utility represents the usefulness of the output compared to the ideal non-private output (as measured for instance by the mean squared error). A simple way to achieve differential privacy is via *output perturbation*, which consists in adding properly calibrated noise to the output of a non-private (potentially deterministic) algorithm. For instance, the Gaussian mechanism relies on the addition of centered Gaussian noise. For completeness, Appendix A recalls a few standard properties and results related to DP. More details can be found in the excellent textbook by Dwork and Roth (2014).

Privacy model for decentralized ML. In the context of privacy-preserving ML in the centralized setting (Chaudhuri et al., 2011), one assumes the presence of a trusted curator

who can collect and process the raw dataset $\mathcal{D}$: the output of an algorithm $\mathcal{A}$ thus solely consists of a machine learning model trained on $\mathcal{D}$. However, in decentralized learning there is no trusted curator and one should thus adapt the privacy model appropriately. In this manuscript, I will generally assume that each user does not trust any other party and seeks to protect against an adversary that may observe everything he/she communicates. In the sense of Definition 1.1, the output of $\mathcal{A}$ will thus consist of the transcript of all messages exchanged by users during training, which is usually referred to as the local model of differential privacy (Kasiviswanathan et al., 2008; Duchi et al., 2013). Note that some relaxations of this strong model will be studied in Chapter 4.

Remark 1.1 (User-level DP)**.** *In the context of decentralized learning where $\mathcal{D} = \cup_u \mathcal{D}_u$ and $\mathcal{D}' = \cup_u \mathcal{D}'_u$, I will sometimes consider a stronger variant of DP by defining the neighboring relation $\mathcal{D} \sim \mathcal{D}'$ to be such that $\mathcal{D}$ and $\mathcal{D}'$ differ on a single user's local dataset. This effectively hides the influence of a user's whole dataset rather than a single of its data points. This variant is sometimes referred to as* user-level DP *(McMahan et al., 2018), as opposed to the classic* record-level DP*. Unless otherwise noted, I will consider record-level DP.*

1.2 Overview of Featured Contributions

In this manuscript, I focus on a representative set of contributions which reflect my current research interests and form a consistent whole. Relevant related work is discussed in the introduction of each chapter.

Chapter 2 is devoted to *decentralized and private learning of personalized models*. The key idea is to tackle the high heterogeneity of local datasets by learning a specific model for each user, rather than a single global model as in (1.2). Inspired by multi-task learning, we model the underlying relationships between the users' tasks as a graph and propose an objective function to learn the relationship graph together with personalized models leveraging this graph. We design fully decentralized and asynchronous algorithms to optimize this objective in a scalable and communication-efficient manner, and analyze their convergence rates. We also propose a differentially private variant to bound the amount of leakage from the messages sent by each user and formally characterize the resulting privacy-utility trade-offs. To the best of our knowledge, our work was the first to propose to learn personalized models for decentralized learning.

The work covered in this chapter was published in a series of three conference papers. **[AB-Conf15]** was done during the Masters internship of P. Vanhaesebrouck that I co-supervised with M. Tommasi. **[AB-Conf13]** was done in collaboration with R. Guerraoui's group at EPFL. Finally, **[AB-Conf9]** was done during a visit of V. Zantedeschi (PhD student at Univ. St-Etienne) at Inria, which I co-supervised with M. Tommasi.

Chapter 3 deals with *decentralized and private learning with pairwise loss functions*, where each term in the objective involves a pair of data points. Such pairwise objectives, commonly used in ML problems like ranking and metric learning, are challenging to solve in the decentralized setting because they do not decompose into a sum of local objective functions as in (1.2). We propose the first efficient decentralized algorithms to learn with pairwise objectives. Our algorithms are based on an iterative data propagation step which allows each user to access data points from other users, and can operate in an asynchronous setting. Interestingly, our convergence results are able to capture some degree of data and network-dependence in the rates. To address privacy constraints raised by data propagation, we propose protocols to estimate pairwise statistics in the local model of differential privacy, from which we can readily obtain private versions of the above decentralized algorithms.

The work covered in this chapter was published in a series of three conference papers. [**AB-Conf21**] and [**AB-Conf16**] were part of the PhD thesis of I. Colin, who I worked with during my postdoc at Télécom Paris. [**AB-Conf3**] was done during a visit of T. Kulkarni (PhD student at Warwick Univ.) at Inria under my supervision, and in collaboration with J. Bell and A. Gascón (The Alan Turing Institute).

Chapter 4 presents two approaches to *achieve better privacy-utility trade-offs for decentralized learning* than what is possible in the local model of differential privacy. Our first contribution focuses on the subproblem of decentralized differentially private averaging, the key primitive needed in decentralized learning with an untrusted aggregator. We propose a protocol in which pairs of users securely exchange correlated Gaussian noise, and show that it can nearly match the privacy-utility trade-off of the trusted curator setting with only logarithmic communication cost per user. We also show how to guarantee the correctness of our protocol in the presence of malicious users by relying on standard cryptographic primitives. Our second contribution is the first work to show that fully decentralized algorithms can formally amplify privacy guarantees. To this end, we introduce a novel relaxation of local DP that naturally arise in the fully decentralized setting and design algorithms operating on ring and complete topologies. For tasks like real summation, discrete histogram computation and learning with stochastic gradient descent, we show that the privacy-utility trade-offs of our algorithms significantly improve upon local DP, in some cases matching the performance of the trusted curator setting.

The work covered in this chapter is available in two recent preprints. [**AB-Preprint7**] was done in collaboration with Inria colleagues J. Ramon and PhD student C. Sabater. [**AB-Preprint6**] was done with E. Cyffers during her Masters internship. She started a PhD under my supervision in October 2021.

The practical relevance of our methods is illustrated by numerical experiments on

synthetic and real data. To conclude the manuscript, **Chapter 5** presents the future I envision for decentralized and privacy-preserving machine learning and describes a short to mid-term research project to address some of the remaining challenges in this area.

1.3 Other Contributions

The intersection of machine learning, privacy and distributed algorithms lies at the heart of my current research interests. I briefly mention here some contributions and activities that are not described in this manuscript to favor conciseness and consistency. I have participated to a collaborative literature survey on decentralized and federated learning [**AB-Journal2**]. I have studied the privacy of decentralized algorithms for rumor spreading [**AB-Conf4**], and how to mitigate leakage from data-dependent communications [**AB-Preprint2**]. Beyond decentralized learning, I have contributed to parallel/distributed ML for cluster computing [**AB-Conf19**]; [**AB-Journal7**]; [**AB-Conf11**] and private optimization in the centralized setting [**AB-Preprint3**]. I have also worked on more applied questions related to privacy and ML. This includes work on speaker anonymization in speech [**AB-Conf10**]; [**AB-Conf8**]; [**AB-Conf7**]; [**AB-Conf5**]; [**AB-Preprint4**] (through the co-supervision of the PhD thesis of B. Srivastava and postdoc of M. Maouche), studying the privacy risks of releasing ML models in genomics [**AB-Conf6**]; [**AB-Journal3**] (in collaboration with researchers at The Alan Turing Institute), and a recent project with Lille University Hospital to deploy decentralized learning for multi-centric medical studies (with co-supervision of an engineer: Y. Bouillard). Finally, I am teaching an MSc-level course on privacy-preserving ML at the University of Lille.[2]

Some of the contributions presented in this manuscript relate to my work on (centralized, non-private) ML. Chapter 3 draws upon my work on learning with pairwise objectives [**AB-Journal9**]; [**AB-Conf23**]; [**AB-Conf14**]; [**AB-Conf11**]; [**AB-Conf2**] (through the co-supervision of the PhD thesis of R. Vogel). The idea of learning pairwise relationships between users in Chapter 2 is related to metric learning, which has been one of the main topics of my early research. I am the co-author of a popular survey [**AB-Preprint10**] and book [**AB-Book1**] on this topic. I am also one of the maintainers of `metric-learn`,[3] an open source Python package for metric learning [**AB-Journal4**]. The development was driven in part by an engineer (W. de Vazelhes) that I supervised, and benefited from collaborations with the `scikit-learn` community.

Finally, I have contributed to several other problems in machine learning, among which scalable kernel methods [**AB-Journal6**], representation learning for NLP [**AB-Conf12**] and fairness in machine learning [**AB-Conf2**]. The full list of my scientific contributions can be found in the bibliography part at the end of the manuscript.

[2] http://researchers.lille.inria.fr/abellet/teaching/private_machine_learning_course.html
[3] https://github.com/scikit-learn-contrib/metric-learn

Chapter 2

Decentralized and Private Learning of Personalized Models

Dealing with data heterogeneity is one of the main challenges in decentralized learning, as discussed in Chapter 1. When learning a global model θ as in (1.1) on highly heterogeneous local datasets, there is typically a lack of consensus in the model updates across users, which can lead to a slow convergence and/or a suboptimal model (Karimireddy et al., 2020; Hsieh et al., 2020). As we have shown in recent work [**AB-Preprint1**], this is even more pronounced in fully decentralized approaches as they rely on local aggregation in sparse topologies. More fundamentally, the very existence of a global model that is accurate for all users is at odds with the heterogeneity of local datasets.

In this chapter, we take a different route to the problem of data heterogeneity in decentralized learning and propose to *learn personalized models* $\theta_1, \ldots, \theta_n$ *for each user* in a collaborative fashion. Our idea is to leverage the fact that in many large-scale applications (e.g., predictive modeling in smartphones apps), each user exhibits distinct behaviors/preferences but is sufficiently similar to some other users to benefit from sharing information with them. We thus propose to discover the relationships between the users' tasks in the form of a sparse *collaboration graph* and learn personalized models that leverage this graph to achieve better generalization performance. Inspired by multi-task learning, we formulate this problem as a joint objective function over the models and the graph and design fully decentralized and asynchronous algorithms to solve it by alternating optimization over the two sets of parameters.

To update the models given a fixed graph, we propose a fast decentralized coordinate descent algorithm which is particularly well-suited to learn linear models. We also propose a decentralized Frank-Wolfe algorithm to learn nonlinear models as combinations of pre-trained models in a boosting fashion, with logarithmic communication complexity. In these algorithms, the collaboration graph serves as an overlay to restrict the communication to pairs of users who appear to be sufficiently similar so as to ensure that our algorithms scale well with the number of users. To update the collaboration

graph given the models, we design a decentralized algorithm in which users update their neighborhood by communicating only with small random subsets of other users obtained through a peer sampling service (Jelasity et al., 2007). Our approach is flexible enough to accommodate various graph regularizers, which allow to easily control the sparsity of the learned graph (and thereby ensure the scalability of the model update step). We provide formal convergence guarantees for all our algorithms under convex loss functions, with explicit rates. Finally, we propose a differentially private version of our approach to guarantee that the messages sent by the users over the network during the execution of the algorithm do not reveal significant information about any individual data point. We provide a formal analysis of the utility loss due to privacy, showing that our formulation naturally tone down the influence of users with small datasets (who propagate more noisy information). The practical relevance of our approach for dealing with heterogeneous local datasets is illustrated by numerical experiments.

This chapter provides a unified treatment of the work published in a series of three conferences papers [**AB-Conf15**]; [**AB-Conf13**]; [**AB-Conf9**].

Related work. Standard federated learning approaches train a unique model for all users (McMahan et al., 2017; Konečný et al., 2016; Li et al., 2020; Karimireddy et al., 2020; Mohri et al., 2019; Hsieh et al., 2020). This is also the case for fully decentralized approaches, which have seen a recent surge of interest (Lian et al., 2017; Jiang et al., 2017; Tang et al., 2018; Lian et al., 2018; Assran et al., 2019; Koloskova et al., 2019; Scaman et al., 2019; Hendrikx et al., 2019; Koloskova et al., 2020). While some of these approaches are designed to mitigate the impact of data heterogeneity on the convergence, none of them studies the interplay with privacy constraints (on this topic, see our recent work [**AB-Preprint5**]). More crucially, learning a single global model may not adequately capture the idiosyncrasies of each user.

To the best of our knowledge, our initial paper [**AB-Conf15**] was the first to learn personalized models in the context of federated/decentralized learning. In work done independently of ours, Smith et al. (2017) proposed a similar formulation, although the task relationships do not form a valid weighted graph and are typically not sparse. Their algorithm relies on a central coordinator: the relations are updated in a centralized way by the server who must regularly access all user models. This can represent a significant communication and computation bottleneck when the number of users is large.

Learning personalized models is now widely regarded as a very effective way to deal with heterogeneous data in federated/decentralized learning. Recent approaches have proposed to derive personalized models from a global "meta-model" (Jiang et al., 2019; Fallah et al., 2020; Khodak et al., 2019), which works only when local distributions are sufficiently close to the global distribution (Deng et al., 2020). Clustered FL (Sattler et al., 2020; Ghosh et al., 2020; Mansour et al., 2020) addresses the potential lack of a global model by assuming that users can be partitioned into several clusters and learning an

independent model for each cluster. Our formulation allows for more nuanced relations among users' models. While our approach is best suited to learn linear models (or linear combinations of pre-trained models), recent approaches (Hanzely and Richtárik, 2020; Hanzely et al., 2020; Dinh et al., 2020) target more general models with nonconvex losses at the cost of considering simpler penalization terms (e.g., the distance to the average model), thereby losing the capability to capture complex relations among users.

While privacy constraints have been studied for decentralized learning of a single global model (see e.g., McMahan et al., 2018; Girgis et al., 2021), we are not aware of other privacy-preserving approaches for learning personalized models.

Outline of the chapter. In Section 2.1, we introduce our formulation for learning personalized models and the fully decentralized setting in which we propose to solve the problem. Section 2.2 presents our decentralized algorithms to learn the models given a fixed collaboration graph. Section 2.3 then introduces an algorithm to update the graph given the models. Section 2.4 shows how to incorporate differential privacy constraints to the proposed approach. Finally, Section 2.5 presents some numerical experiments.

2.1 Proposed Formulation

As introduced in Chapter 1, each user $u \in [\![n]\!]$ holds a local dataset $\mathcal{D}_u$ of m_u data points, which we assume to be drawn from a distribution specific to user u. The true goal of user u is to learn a model parameterized by $\theta_u \in \mathcal{C}$ which generalizes well to unseen data points drawn from its personal distribution. Instead of learning a single global model as in (1.1), which could perform arbitrarily bad for some users, we introduce a new decentralized learning approach where users learn *personalized models* by leveraging relationships between the users' tasks that are learned together with the models.

Objective function. We assume that all users learn models from the same hypothesis class. Since users have datasets of different sizes we introduce a notion of "confidence" $c_u \in \mathbb{R}^+$ for each user u, which should be thought of as proportional to m_u (in practice we simply set $c_u = m_u / \max_{v \in [\![n]\!]} m_v$). The relationships between users are encoded in the form of an undirected and weighted *collaboration graph* $\mathcal{G}_w = ([\![n]\!], w)$ in which nodes correspond to users and nonnegative edge weights $w_{u,v} \geq 0$ reflect the similarity between the learning tasks of users u and v, with $w_{u,v} = 0$ indicating the absence of edge.

We propose to learn the personalized classifiers $\Theta = (\theta_1, \ldots, \theta_n)$ and collaboration graph w by minimizing the following joint optimization problem:

$$\underset{\substack{\Theta \in \mathcal{C}^n \\ w \in \mathcal{W}}}{\arg\min} \left\{ J(\Theta, w) = \sum_{u=1}^{n} d_u(w) c_u F_u(\theta_u; \mathcal{D}_u) + \frac{\lambda_1}{2} \sum_{1 \leq u < v \leq n} w_{u,v} \|\theta_u - \theta_v\|^2 + \lambda_2 g(w) \right\}, \quad (2.1)$$

where $\mathcal{C}$ and $\mathcal{W} = \{w \in \mathbb{R}^{n(n-1)/2} : w \geq 0\}$ are the feasible domains for the models and the graph respectively, $d(w) = (d_1(w), \ldots, d_n(w)) \in \mathbb{R}^n$ is the degree vector with $d_u(w) = \sum_{v=1}^n w_{u,v}$, and $\lambda_1, \lambda_2 \geq 0$ are trade-off hyperparameters.

The joint objective function $J(\Theta, w)$ in (2.1) is composed of three terms. The first one is a (weighted) sum of local objective functions $F_1(\theta_1; \mathcal{D}_1), \ldots, F_n(\theta_n; \mathcal{D}_n)$, each involving only the personal model θ_u and local dataset $\mathcal{D}_u$ of a single user. The second term involves both the models and the graph: it enables collaboration by encouraging two users u and v to have a similar model for large edge weight $w_{u,v}$. This principle, known as graph regularization, is well-established in the multi-task learning literature (Evgeniou and Pontil, 2004; Maurer, 2006). Importantly, the factor $d_u(w)c_u$ in front of the local objective F_u of each user u implements a useful inductive bias: users with larger datasets (large confidence) will tend to connect to other nodes as long as their local objective remains small so that they can positively influence their neighbors, while users with small datasets (low confidence) will tend to disregard their local objective and rely more on information from other users. Finally, the last term $g(w)$ introduces some regularization on the graph weights w used to avoid degenerate solutions (e.g., edgeless graph) and control structural properties such as sparsity (see Section 2.3 for a concrete example). Note that we do not enforce the graph to be connected: different connected components can be seen as modeling clusters of unrelated users.

Remark 2.1 (Flexibility of the formulation)**.** *Our objective* (2.1) *allows for very flexible notions of relationships between the users' tasks. For instance, as $\lambda_1 \to +\infty$ the problem becomes equivalent to learning a shared model for all users in the same connected component of the graph, by minimizing the sum of local objectives independently in each component. On the other hand, setting $\lambda_1 = 0$ corresponds to having each user u learn its classifier θ_u based on its local dataset only (no collaboration). Intermediate values of λ_1 let each user learn its own personal model with the models of other (strongly connected) users acting as a regularizer.*

Decentralized setting. We aim to solve (2.1) problem in a fully decentralized way without relying on a central coordinator. We consider the *asynchronous time model* (Boyd et al., 2006): each user regularly becomes active at the ticks of an independent local clock which follows a Poisson distribution. This is in contrast to the synchronous model where users wake up jointly according to a global clock (and thus need to wait for everyone to finish each round). As local clocks are i.i.d., we can equivalently consider a single clock which ticks when one of the local clocks ticks. This provides a convenient way to state and analyze the algorithms in terms of a global clock counter t.

We assume that each user can send messages to any other user (like on the Internet). However, in order to scale to a large number of users and achieve fruitful collaboration, we will generally restrict the message exchanges to pairs of users whose tasks are most similar according to the collaboration graph $\mathcal{G}_w$. In other words, we use $\mathcal{G}_w$ as a seman-

Algorithm 2.1 Decentralized alternating optimization procedure to solve (2.1).

Input: Dataset $\mathcal{D}_u$ for each user $u \in [\![n]\!]$; Number of alternating steps T

1: Initialize personalized models $\Theta = (\theta_1, \ldots, \theta_n) \in \mathcal{C}^n$ and graph weights $w \in \mathcal{W}$
2: **for** $t = 1$ to T **do**
3: Update Θ given w, with Algorithm 2.2 or Algorithm 2.3
4: Update w given Θ, with Algorithm 2.4
5: **return** Θ, w

tic overlay on the communication layer: a user k can only send messages to its direct neighbors $\mathcal{N}_u = \{v : w_{u,v} > 0\}$ in $\mathcal{G}_w$. Recall that in our approach, the collaboration graph is not known beforehand and will iteratively evolve. Therefore, when updating the collaboration graph, we will also allow users to communicate with a small random set of peers. This can be implemented via a classic distributed systems primitive called a peer sampling service (Jelasity et al., 2007).

Alternating optimization. In general, (2.1) will not be jointly convex in Θ and w, but will be bi-convex if the local objectives and graph regularizer are convex. We thus propose to solve (2.1) by *alternating optimization*, as shown in Algorithm 2.1). The idea is to iterate between updating the models Θ given the current graph w (Section 2.2) and updating the graph given the current models (Section 2.3). Alternating optimization converges to a local optimum under technical conditions, see (Tseng, 2001; Tseng and Yun, 2009; Razaviyayn et al., 2013).

2.2 Learning Personalized Models for Fixed Graph

In this section, we propose decentralized algorithms for solving (2.1) for *fixed graph weights* w. Throughout this section, we will denote the corresponding objective function by $M(\Theta) = J(\Theta, w)$ and drop the explicit dependence of the degrees on w, writing d_u instead of $d_u(w)$ for conciseness.

We first propose a decentralized coordinate descent algorithm which is most useful to learn linear models. Then, we propose a decentralized Frank-Wolfe algorithm that can learn nonlinear models as sparse combinations of base classifiers in a boosting-like manner with logarithmic communication complexity. Both algorithms rely on broadcast-based communication (Aysal et al., 2009; Nedic, 2011), where users send messages to all their neighbors in $\mathcal{G}_w$ at once without expecting a reply.

2.2.1 Decentralized Coordinate Descent Algorithm

In this section, we present and analyze a decentralized algorithm based on Coordinate Descent (CD) techniques (Richtárik and Takác, 2014; Wright, 2015).

Algorithm 2.2 Decentralized coordinate descent algorithm to learn personalized models.

Input: Dataset $\mathcal{D}_u$ for each user $u \in [\![n]\!]$; Graph weights $w \in \mathcal{W}$; Number of iterations T

Each user $u \in [\![n]\!]$ initializes its local model $\theta_u(0) \in \mathbb{R}^p$
for $t = 1$ to T **do**
  $u \leftarrow$ random user from $[\![n]\!]$ `// a random user wakes up`
  User u generates a new version $\theta_u(t)$ of its local model via the CD update (2.3)
  User u sends $\theta_u(t)$ to each user $v \in \mathcal{N}_u$
  for each user $v \neq u$ **do**
    $\theta_v(t) \leftarrow \theta_v(t-1)$ `// other models remain unchanged`
return Each user has $\theta_u(T)$

Setup and notations. In this part, we assume that $\mathcal{C} = \mathbb{R}^p$ for some integer $p \geq 1$, hence for all $u \in [\![n]\!]$ we have $\theta_u \in \mathbb{R}^p$ and $\Theta = (\theta_1, \ldots, \theta_n) \in \mathbb{R}^{np}$. We also make a number of standard assumptions on the regularity and curvature of the objective (see Appendix B for formal definitions of these notions). For any $u \in [\![n]\!]$, we assume that the local objective function F_u of user u is convex in its first argument with L_u^{loc}-Lipschitz continuous gradient. This implies that M is convex in Θ.[1] If we further assume that each F_u is μ_u^{loc}-strongly convex with $\mu_u^{loc} > 0$ (which is the case for instance when F_u includes ℓ_2-regularization), then M is μ-strongly convex with $\mu \geq \min_{1 \leq u \leq n}[d_u c_u \mu_u^{loc}] > 0$. The partial derivative of $M(\Theta)$ w.r.t. the variables θ_u is given by

$$[\nabla M(\Theta)]_u = d_u(\lambda_1 \theta_u + c_u \nabla F_u(\theta_u; \mathcal{D}_u)) - \lambda_1 \sum_{v \in \mathcal{N}_u} w_{u,v} \theta_v. \tag{2.2}$$

For $u \in [\![n]\!]$, we denote by $L_u = d_u(\lambda_1 + c_u L_u^{loc})$ the block Lipschitz constant L_u of $\nabla M(\Theta)$ with respect to block θ_u. We let $L_{min} = \min_u L_u$ and $L_{max} = \max_u L_u$.

Decentralized CD algorithm. We propose a decentralized coordinate descent algorithm to minimize M (Algorithm 2.2). First, each user u initializes its local model $\theta_u(0) \in \mathbb{R}^p$. Then, at time step t, a random user u wakes up and performs two consecutive actions:

- *Update step*: user u updates its local model based on the most recent information $\theta_v(t)$ received from its neighbors $v \in \mathcal{N}_u$:

$$\begin{aligned} \theta_u(t) &= \theta_u(t-1) - (1/L_u)[\nabla M(\Theta(t-1))]_u \\ &= (1-\alpha)\theta_u(t-1) + \alpha\Big(\sum_{v \in \mathcal{N}_u} \frac{w_{u,v}}{d_u}\theta_v(t-1) - \frac{c_u}{\lambda_1}\nabla F_u(\theta_u(t-1); \mathcal{D}_u)\Big), \end{aligned} \tag{2.3}$$

  where $\alpha = \frac{\lambda_1}{\lambda_1 + c_u L_u^{loc}} \in (0, 1]$.

- *Communication step*: user u sends its updated model $\theta_u(t)$ to its neighborhood $\mathcal{N}_u$.

[1] This follows from the fact that the second term in (2.1) is a Laplacian quadratic form, hence convex in Θ.

The update step (2.3) consists in a block coordinate descent update with respect to θ_u and only requires user u to know the models $\theta_v(t-1)$ previously broadcast by its neighbors $v \in \mathcal{N}_u$. Note that the user does not need to know the global iteration counter t, hence no global clock is needed. Algorithm 2.2 is thus fully decentralized and asynchronous. Interestingly, notice that this block CD update is adaptive to the confidence level of each user in two respects: (i) globally, the more confidence, the more importance given to the gradient of the local objective F_u compared to the neighbors' models, and (ii) locally, when $\theta_u(t-1)$ is close to a minimizer of F_u (which is the case for instance if we initialize $\theta_u(0)$ to such a minimizer), users with low confidence will trust their neighbors' models more aggressively than users with high confidence (which will make more conservative updates).[2] This is in line with the intuition that users with low confidence should diverge more quickly from their local minimizer than those with high confidence.

Convergence analysis. Under our assumption that the local clocks of users are i.i.d., the above algorithm can be seen as a randomized block coordinate descent algorithm (Wright, 2015). It enjoys a fast linear convergence rate when M is strongly convex.

Proposition 2.1. *For $T > 0$, let $(\Theta(t))_{t=1}^{T}$ be the sequence of iterates generated by Algorithm 2.2 running for T iterations from an initial point $\Theta(0) \in \mathbb{R}^{np}$. Let $M^\star \in \min_{\Theta \in \mathbb{R}^{np}} M(\Theta)$. When M is μ-strongly convex, we have:*

$$\mathbb{E}\left[M(\Theta(T)) - M^\star\right] \leq \left(1 - \frac{\mu}{nL_{max}}\right)^T \left(M(\Theta(0)) - M^*\right).$$

Proof. This follows from a slight adaptation of the proof of Wright (2015, Theorem 1 therein) to the block coordinate descent case. □

Remark 2.2. *For general convex M, an $O(1/T)$ rate can be obtained, see Wright (2015).*

Proposition 2.1 shows that each iteration shrinks the suboptimality gap by a constant factor. While this factor degrades linearly with the number of users n, this is compensated by the fact that the number of iterations done in parallel also scales roughly linearly with n (because users operate asynchronously and in parallel). We thus expect the algorithm to scale gracefully with the size of the network if the number of updates per user remains constant. The value $\frac{\mu}{L_{max}} \geq \frac{\min_{1\leq u\leq n}[d_u c_u \mu_u^{loc}]}{\max_{1\leq u\leq n}[d_u(\lambda_1 + c_u L_u^{loc})]} > 0$ is the ratio between the lower and upper bound on the curvature of M. Focusing on the relative differences between users and assuming constant μ_u^{loc}'s and L_u^{loc}'s, it indicates that the algorithm converges faster when the degree-weighted confidence of users is approximately the same. On the other hand, two situations can represent a bottleneck for convergence: (i) a high-confidence and high-degree user (the progress is then very dependent on the updates of that particular user), and (ii) a low-confidence, poorly connected user (hence converging slowly).

[2] This second property is in contrast to a (centralized) gradient descent approach which would use a constant, more conservative step size (equal to the global Lipschitz constant of M) for all users.

Remark 2.3 (ADMM-based algorithm)**.** *In [**AB-Conf15**], we introduced an algorithm based on asynchronous decentralized ADMM (Wei and Ozdaglar, 2013) by reformulating the problem as a partial consensus over neighbors' models. The algorithm relies on gossip communication (i.e., bidirectional exchanges between pairs of users) and is more general than decentralized CD (e.g., it does not assume that the local objectives are smooth). However, its convergence rate is slower in theory and in practice in the strongly convex case (see [**AB-Conf13**] for an empirical comparison).*

2.2.2 Communication-Efficient Decentralized Frank-Wolfe Algorithm

As an alternative to the previous algorithm, in this section we propose a decentralized Frank-Wolfe algorithm for learning personalized nonlinear classifiers in a boosting manner with logarithmic communication complexity in the number of model parameters.

Setup and notations. For simplicity, we focus on binary classification where each data point $x \in \mathcal{X}$ is associated with a binary label $y \in \{-1, 1\}$. We propose that each user $u \in [\![n]\!]$ learns a personal classifier as a weighted combination of a set of K real-valued base predictors $H = \{h_k : \mathcal{X} \to \mathbb{R}\}_{k=1}^K$, i.e., a mapping $x \mapsto \operatorname{sign}(\sum_{k=1}^K [\theta_u]_k h_k(x))$ parameterized by $\theta_u \in \mathbb{R}^K$. The base predictors may be weak classifiers as in standard boosting, or stronger predictors pre-trained on separate data (e.g., public, crowdsourced, or collected from users who opted in to share personal data). We denote by $A_u \in \mathbb{R}^{m_u \times K}$ the matrix whose (i,k)-th entry gives the margin achieved by the k-th base classifier on the i-th data point (x_i, y_i) in $\mathcal{D}_u$, so that for $i \in [\![m_u]\!]$, $[A_u\theta_u]_i = y_i \sum_{k=1}^K [\theta_u]_k h_k(x_i)$ gives the margin achieved by the personalized combination θ_u. Only user u has access to A_u.

Drawing inspiration from ℓ_1-Adaboost (Shen and Li, 2010; Wang et al., 2015), we instantiate the local objective $F_u(\theta_u; \mathcal{D}_u)$ for each user u as follows:

$$F_u(\theta_u; \mathcal{D}_u) = \log\Big(\sum_{i=1}^{m_u} \exp(-[A_u\theta_u]_i)\Big), \tag{2.4}$$

and let the feasible domain $\mathcal{C} = \{\theta \in \mathbb{R}^K : \|\theta\|_1 \le \beta\}$ where $\beta \ge 0$ is a hyperparameter to favor sparse models by controlling their ℓ_1-norm. Note that with the log loss function (2.4) the objective $M(\Theta) = J(\Theta, w)$ is convex and continuously differentiable, and the domain $\mathcal{C}^n$ is a compact and convex subset of $(\mathbb{R}^K)^n$. The partial derivative of $M(\Theta)$ w.r.t. the variables θ_u is given by

$$\nabla[M(\Theta)]_u = -d_u c_u \pi_u^\top A_u + \lambda_1\Big[d_u\theta_u - \sum_{v=1}^n w_{u,v}\theta_v\Big], \text{ with } \pi_u = \frac{\exp(-A_u\theta_u)}{\sum_{i=1}^{m_u}\exp(-A_u\theta_u)_i}. \tag{2.5}$$

Decentralized FW algorithm. We propose a decentralized algorithm based on Frank-Wolfe (FW), also known as conditional gradient descent (Frank and Wolfe, 1956; Clarkson, 2010; Jaggi, 2013). Our approach, shown in Algorithm 2.3, proceeds as follows. Each

Algorithm 2.3 Decentralized Frank-Wolfe algorithm to learn personalized models.

Input: Dataset $\mathcal{D}_u$ for each user $u \in [\![n]\!]$; Graph weights $w \in \mathcal{W}$; Number of iterations T; Step sizes $(\gamma(t))_{t\geq 1}$

Each user $u \in [\![n]\!]$ initializes its local model $\theta_u(0) \in \mathcal{C}$
for $t = 1$ to T **do**
  $u \leftarrow$ random user from $[\![n]\!]$ `// a random user wakes up`
  User u generates a new version $\theta_u(t)$ of its local model via the FW update (2.6)
  User u sends $\theta_u(t)$ to each user $v \in \mathcal{N}_u$
  for each user $v \neq u$ **do**
    $\theta_v(t) \leftarrow \theta_v(t-1)$ `// other models remain unchanged`
return Each user has $\theta_u(T)$

personal classifier is initialized to some feasible point $\theta_u(0) \in \mathcal{C}$ (such as the zero vector). Then, at each step t, a random user u wakes up and performs the following actions:

1. *Update step:* user u performs a FW update on its local model based on the most recent information $\theta_v(t-1)$ received from its neighbors $v \in \mathcal{N}_u$:

$$\theta_u(t) = (1-\gamma(t))\theta_u(t-1) + \gamma(t)\, s_u(t), \tag{2.6}$$

   where $\gamma(t) = \frac{2n}{t+2n}$ and $s_u(t) = \beta\,\mathrm{sign}(-(\nabla[M(\Theta(t-1))]_u)_{k_u(t)})e_{k_u(t)}$, with $k_u(t) = \arg\max_k[|\nabla[M(\Theta(t-1))]_u|]_k$ and $e_{k_u(t)}$ the unit vector with 1 in the $k_u(t)$-th entry.

2. *Communication step:* user u sends its updated model $\theta_u(t)$ to its neighborhood $\mathcal{N}_u$.

By inspecting the form of the partial gradient $\nabla[M(\Theta(t-1))]_u$ in (2.5), we see that the proposed update (2.6) has an intuitive interpretation. First, it preserves the flavor of classical boosting by incorporating a single base classifier at a time: the one performing best on the local dataset $\mathcal{D}_u$ of user u reweighted by π_u (i.e., points that are poorly classified get more importance). On the other hand, it incorporates an additional bias towards selecting base predictors that are popular amongst neighbors in the collaboration graph. The relative importance of the two terms is ruled by the user confidence c_u.

As in our previously introduced decentralized CD algorithm (Algorithm 2.2), the update (2.6) only requires the knowledge of the models of neighboring users. Since the number of neighbors is typically small, updates can occur in parallel in different parts of the network, ensuring that the procedure scales well with the number of users.

Convergence analysis. The above algorithm achieves an $O(1/t)$ convergence rate, as shown by the following result.

Proposition 2.2 ([AB-Conf9]). *Algorithm 2.3 takes at most $6n(C_M^{\otimes} + p_0)/\varepsilon$ iterations to find an approximate solution $\Theta \in \mathcal{C}^n$ that satisfies, in expectation, $M(\Theta) - M(\Theta^*) \leq \varepsilon$, where $C_f^{\otimes} \leq 4\beta^2 \sum_{u=1}^n d_u(c_u\|A_u\|^2 + \lambda_1)$ and $p_0 = M(\Theta(0)) - M(\Theta^*)$.*

Sketch of proof. We essentially follow the proof technique proposed by Jaggi (2013) and refined by Lacoste-Julien et al. (2013) for the case of block coordinate Frank-Wolfe. It is based on defining a surrogate for the optimality gap $M(\Theta) - M(\Theta^*)$, where $\Theta^* \in \arg\min_{\Theta \in \mathcal{M}} M(\Theta)$. Under an appropriate notion of smoothness for M over the feasible domain, the convergence is established by showing that the gap decreases in expectation with the number of iterations, because at a given iteration t the block-wise surrogate gap at the current solution is minimized by the greedy update $s_u(t)$. □

Proposition 2.2 shows that large degrees for users with low confidence and small margins penalize the convergence rate less than users with large confidence and large margins. This is rather intuitive as users in the latter case have greater influence on the overall objective (2.1).

Communication and memory costs. Remarkably, using a few tricks in the representation of the sparse updates, the communication and memory cost needed by our algorithm to converge to an approximate solution of arbitrary precision can be shown to be linear in the number of edges of the graph and *logarithmic* in the number of base predictors (see **[AB-Conf9]** for details). For the classic case where base predictors consist of a constant number of decisions stumps per feature, this translates into a logarithmic cost in the dimensionality of the data leading to significantly better complexities than the coordinate descent algorithm introduced in Section 2.2.1 (see **[AB-Conf9]** for experimental results).

Remark 2.4 (Other loss functions)**.** *While we focused on the Adaboost-style loss* (2.4) *for its nice connection to boosting and its ability to learn nonlinear models in a convex formulation, the proposed algorithm and analysis readily extend to other convex loss functions.*

2.3 Learning the Graph for Fixed Models

In the previous section, we have proposed and analyzed two algorithms to learn the model parameters Θ given a fixed collaboration graph w. To make our fully decentralized alternating optimization scheme complete, we now turn to the converse problem of optimizing the graph weights w given fixed models Θ. We will work with flexible graph regularizers $g(w)$ that are *weight and degree-separable*:

$$g(w) = \sum_{u<v} g_{u,v}(w_{u,v}) + \sum_{u=1}^{n} g_u(d_u(w)),$$

where $g_{u,v} : \mathbb{R} \to \mathbb{R}$ and $g_u : \mathbb{R} \to \mathbb{R}$ are convex and smooth. This allows to regularize weights and degrees in a flexible way,[3] while maintaining a separable structure which is

[3]For instance, it encompasses some regularizers recently introduced in the graph signal processing community (Dong et al., 2016; Kalofolias, 2016; Berger et al., 2018).

Algorithm 2.4 Decentralized algorithm to learn the collaboration graph.

Input: Dataset $\mathcal{D}_u$ for each user $u \in [\![n]\!]$; Models $\Theta = (\theta_1, \ldots, \theta_n) \in \mathcal{C}^n$; Number of iterations T

```
Initialize graph weights w(0) ∈ W
for t = 1 to T do
    u ← random user from [[n]]                          // a random user wakes up
    V ← random set of ρ users                           // obtained by peer sampling
    User u requests θ_v, F_v(θ_v; D_v) and d_v(w) from each user v ∈ V
    User u updates its weights w(t)_{u,V} via the update (2.8)
    For each v ∈ V, user u sends w(t)_{u,v} to v
    for each pair of users z, z' ∉ V ∪ {u} do
        w(t)_{z,z'} ← w(t−1)_{z,z'}                     // other weights remain unchanged
return Graph weights w(T)
```

key to the design of an efficient decentralized algorithm. We denote the graph learning objective function by $G(w) = J(\Theta, w)$ for fixed models Θ. Note that $G(w)$ is convex in w.

Decentralized algorithm. Our goal is to design a fully decentralized algorithm to update the collaboration graph $\mathcal{G}_w$. We thus need users to communicate beyond their current direct neighbors in $\mathcal{G}_w$ to discover new relevant neighbors. In order to preserve scalability, a user will only communicate with small random batches of other users. In a decentralized system, this can be implemented by a classic primitive known as a peer sampling service (Jelasity et al., 2007; Kermarrec et al., 2011). The peer sampling services allows a user u wakes up to sample uniformly and without replacement a set $\mathcal{V}$ of $\rho \in [\![n-1]\!]$ users from the set $[\![n]\!] \setminus \{u\}$. We will denote by $w_{u,\mathcal{V}}$ the ρ-dimensional subvector of a vector $w \in \mathbb{R}^{n(n-1)/2}$ corresponding to the entries $\{(u,v)\}_{v \in \mathcal{V}}$. Let $\Delta_{u,\mathcal{V}} = (\|\theta_u - \theta_v\|^2)_{v \in \mathcal{V}}$, $p_{u,\mathcal{V}} = (c_u F_u(\theta_u; \mathcal{D}_u) + c_v F_v(\theta_v; \mathcal{D}_v))_{v \in \mathcal{V}}$ and $v_{u,\mathcal{V}}(w) = (g'_u(d_u(w)) + g'_v(d_v(w)) + g'_{u,v}(w_{u,v}))_{v \in \mathcal{V}}$. The partial derivative of the objective $G(w)$ with respect to the variables $w_{u,\mathcal{V}}$ can be written as follows:

$$[\nabla G(w)]_{u,\mathcal{V}} = p_{u,\mathcal{V}} + (\lambda_1/2)\Delta_{u,\mathcal{V}} + \lambda_2 v_{u,\mathcal{V}}(w). \tag{2.7}$$

We denote by $L_{u,\mathcal{V}}$ is the Lipschitz constant of ∇G with respect to block $w_{u,\mathcal{V}}$.

Our decentralized algorithm, given in Algorithm 2.4, works as follows. We start from some arbitrary weight vector $w(0) \in \mathcal{W}$, each user having a local copy of its $n-1$ weights. At each time step t, a random user u wakes up and performs the following actions:

1. Draw a set $\mathcal{V}$ of ρ users and request their current models, loss value and degree.

2. Update the associated weights:

$$w(t)_{u,\mathcal{V}} \leftarrow \max\big(0, w(t-1)_{u,\mathcal{V}} - (1/L_{u,\mathcal{V}})[\nabla G(w(t-1))]_{u,\mathcal{V}}\big). \tag{2.8}$$

3. Send each updated weight $w(t)_{u,v}$ to the associated user in $v \in \mathcal{V}$.

Again, Algorithm 2.4 is fully decentralized: the information requested at step 1 above is sufficient to compute (2.7) and thereby (2.8). Therefore, updates can occur asynchronously and in parallel.

Convergence, communication and memory. For the case where g is strongly convex, we obtain the following linear convergence rate. Note that in for the general convex case, we can obtain a slower $O(1/T)$ convergence rate.

Theorem 2.1 ([AB-Conf9]). *Assume that the graph regularizer $g(w)$ is μ-strongly convex. Let $T > 0$ and G^* be the optimal objective value and $L_{max} = \max_{(u,\mathcal{V})} L_{u,\mathcal{V}}$. The iterates $(w(t))_{t=1}^T$ of Algorithm 2.4 satisfy:*

$$\mathbb{E}[G(w(T)) - G^*] \leq \Big(1 - \frac{2\rho\mu}{n(n-1)L_{max}}\Big)^T (G(w(0)) - G^*).$$

Sketch of proof. We show that Algorithm 2.4 can be seen as an instance of proximal coordinate descent (PCD) (Tseng and Yun, 2009; Richtárik and Takác, 2014) on a slightly modified objective function. Unlike the standard PCD setting which focuses on disjoint blocks, our coordinate blocks exhibit a specific overlapping structure that arises as soon as $\rho > 1$ (as each weight is shared by two users). We build upon the PCD analysis due to (Wright, 2015), which we adapt to account for our overlapping block structure. □

Theorem 2.1 shows that our algorithm cuts the expected suboptimality gap by a constant factor at each iteration. This convergence rate is of the same order as the one for learning the models with our decentralized CD algorithm (Proposition 2.1) and typically faster than the sublinear rate of our decentralized FW approach (Proposition 2.2). This suggests to run a small number of graph updates per user before re-updating the models given the new graph. We note that the parameter ρ rules a trade-off between communication/memory costs and convergence rate, see **[AB-Conf9]** for details.

Example graph regularizer. In our experiments, we use a graph regularizer inspired from (Kalofolias, 2016), defined as $g(w) = \lambda_3\|w\|^2 - \mathbf{1}^\top \log(d(w))$ with $\lambda_3 > 0$. The log term ensures that all nodes have nonzero degrees without preventing the graph from having several connected components. This choice of regularizer has a number of advantages. First, λ_3 provides a direct way to tune the sparsity of the graph: the smaller λ_3, the more concentrated the weights of a given user on the users with the closest models. This allows us to control the trade-off between accuracy and communication in the model update step (Section 2.2). The resulting objective is also strongly convex and block-Lipschitz continuous[4] and thus matches the assumptions of Theorem 2.1. Finally, as discussed by Kalofolias (2016), tuning the importance of the log-degree term with respect to the other graph terms has simply a scaling effect, thus we can simply set $\lambda_2 = \lambda_1$ in (2.1).

[4]In practice we add a small positive constant inside the log to make it smooth (Koriche, 2018).

Remark 2.5. *To reduce the number of variables to optimize, each user can keep to* 0 *the weights corresponding to users whose current model is most different to theirs. This heuristic has a negligible impact on the solution quality in sparse regimes (small* λ_3*).*

2.4 Incorporating Differential Privacy Constraints

In the previous sections, we have proposed an alternating optimization approach to jointly learn the personalized models and the collaboration graph. While there is no direct exchange of data between users, the sequence of iterates broadcast by a user may reveal information about its private dataset through the gradient of the local objective.

In this section, we propose a differentially private version of our approach. We first define the privacy model, then introduce a differentially private variant of our decentralized coordinate descent algorithm of Section 2.2.1. Finally, we show how differential privacy guarantees can be obtained for the full alternating optimization procedure.

2.4.1 Privacy Model

As explained in Chapter 1, we aim to protect users from an eavesdropping adversary who can observe all the information sent over the network during the execution of the algorithm (but cannot access the users' internal memory). Note that this covers the case where $n-1$ users collude to learn information about the remaining user. This is a very strong privacy model known as local DP (Kasiviswanathan et al., 2008; Duchi et al., 2013): each user does not trust anyone to process his/her data, hence the privacy-preserving mechanism must be implemented locally by each user. Some general relaxations of this strong model will be studied in Chapter 4.

We want to ensure that the adversary cannot learn much information about any individual data point of any user's dataset. We instantiate this using differential privacy, adapting it to our context. Following the notations in Definition 1.1, we view each user $u \in [\![n]\!]$ as running an algorithm $\mathcal{A}_u(\mathcal{D}_u)$ which takes as input its local dataset $\mathcal{D}_u$ and outputs all the information sent by u over the network during the execution of the algorithm. As our approach is based on an alternating procedure between updating the models and updating the graphs, the output of $\mathcal{A}_u(\mathcal{D}_u)$ includes in particular the sequence of model parameters and graph weights broadcast by the user across all alternating optimization steps. Given the privacy budget (ϵ_u, δ_u) of each user $u \in [\![n]\!]$, our goal is to ensure that $\mathcal{A}_u(\mathcal{D}_u)$ satisfies (ϵ_u, δ_u)-DP for all users simultaneously.

2.4.2 Privacy-Preserving Decentralized Coordinate Descent

The critical step for privacy is the model update step. Here, we propose and analyze a differentially private version of our decentralized coordinate descent algorithm pre-

sented in Section 2.2.1 (Algorithm 2.2). Our approach is based on the standard Laplace mechanism (see Appendix A). Formally, it consists in replacing the update step (2.3) by:

$$\widetilde{\theta}_u(t) = (1-\alpha)\widetilde{\theta}_u(t-1) + \alpha\Big(\sum_{v\in\mathcal{N}_u} \frac{w_{u,v}}{d_u}\widetilde{\theta}_v(t-1) - \frac{c_u}{\lambda_1}\big(\nabla F_u(\widetilde{\theta}_u(t-1);\mathcal{D}_u) + \eta_u(t)\big)\Big), \quad (2.9)$$

where $\eta_u(t) \sim \mathrm{Lap}(s_u(t))^p \in \mathbb{R}^p$ is a noise vector drawn from a centered Laplace distribution with finite scale $s_u(t) \geq 0$.[5] We allow the noise to potentially depend on the global iteration number t. Note that the only difference with the non-private update (2.3) is that user u adds appropriately scaled Laplace noise to the gradient of its local loss F_u. The rest of the algorithm proceeds as in the non-private version described in Algorithm 2.2: in particular, user u sends the resulting (noisy) iterate $\widetilde{\theta}_u(t)$ to its neighbors.

Privacy guarantees. Assume that update (2.9) is run T_u times by user u within the total $T > 0$ iterations across the network. Let $\mathcal{T}_u = \{t_u^k\}_{k=1}^{T_u}$ be the set of iterations at which user u woke up and consider the mechanism $\mathcal{A}_u^{\mathrm{CD}}(\mathcal{D}_u) = \{\widetilde{\theta}_u(t_u) : t_u \in \mathcal{T}_u\}$. The following result shows how to scale the noise at each iteration so as to provide the desired overall differential privacy guarantees.

Proposition 2.3 (Privacy guarantee for $\mathcal{A}_u^{\mathrm{CD}}$ **[AB-Conf13]**). *Let $u \in [\![n]\!]$ and assume that $F_u(\theta;\mathcal{D}_u) = \frac{1}{m_u}\sum_{x\in\mathcal{D}_u} f(\theta;x)$ where $f(\cdot;x)$ is L_0-Lipschitz with respect to the ℓ_∞-norm for all $x \in \mathcal{X}$. For any $t_u \in \mathcal{T}_u$, let $s_u(t_u) = \frac{2L_0}{\epsilon_u(t_u)m_u}$ for some $\epsilon_u(t_u) > 0$. Given an initial point $\widetilde{\Theta}(0) \in \mathbb{R}^{np}$ independent of $\mathcal{D}_u$, the mechanism $\mathcal{A}_u^{CD}(\mathcal{D}_u)$ is ϵ_u-DP with $\epsilon_u = \sum_{t_u=1}^{T_u}\epsilon_u(t_u)$.*

Sketch of proof. For $t_u \in \mathcal{T}_u$, we first study the base mechanism which releases a single iterate $\widetilde{\theta}_u(t_u)$. We bound the sensitivity of the non-private update rule (2.3) using the fact that the Lipschitz property of f implies that its gradients have ℓ_1-norm bounded by L_0 (see Lemma B.1), and show that the noise scale $s_u(t_u)$ is sufficient to ensure $\epsilon_u(t_u)$-DP according to the Laplace mechanism (see Theorem A.4). The DP guarantee for $\mathcal{A}_u$ then follows from simple composition. □

Remark 2.6. *We can obtain a similar (ϵ,δ)-DP result if we assume L_0-Lipschitzness of f w.r.t. ℓ_2-norm (instead of ℓ_1) and use the Gaussian mechanism (instead of the Laplace mechanism).*

For simplicity, Proposition 2.3 uses the simple composition property of DP. A sublinear scaling of ϵ_u with respect to T_u can be obtained under approximate (ϵ,δ)-DP using advanced composition theorems (see Appendix A). In any case, the noise scale needed to guarantee DP for a user u is inversely proportional to the size m_u of its local dataset $\mathcal{D}_u$. While this is often the case in differentially private machine learning, it is especially appealing in our formulation as the confidence weights c_u's tune down the importance of users with small datasets (preventing their overly noisy information to spread) and give more importance to users with larger datasets (who propagate useful information).

[5] We use the convention $\mathrm{Lap}(0) = 0$ w.p. 1.

Utility guarantees. The next result quantifies how the added noise affects the convergence of the algorithm and illustrates our previous point.

Theorem 2.2 (Utility loss [AB-Conf13]). *For any $T > 0$, let $(\widetilde{\Theta}(t))_{t=1}^{T}$ be the sequence of iterates generated by T iterations of update (2.9) from an initial point $\widetilde{\Theta}(0) \in \mathbb{R}^{np}$. For μ-strongly convex M, we have:*

$$\mathbb{E}\left[M(\widetilde{\Theta}(T)) - M^\star\right] \leq \left(1 - \frac{\mu}{nL_{max}}\right)^T \left(M(\widetilde{\Theta}(0)) - M^\star\right) + \frac{1}{nL_{min}} \sum_{t=0}^{T-1} \sum_{u=1}^{n} \left(1 - \frac{\mu}{nL_{max}}\right)^t \left(d_u c_u s_u(t)\right)^2.$$

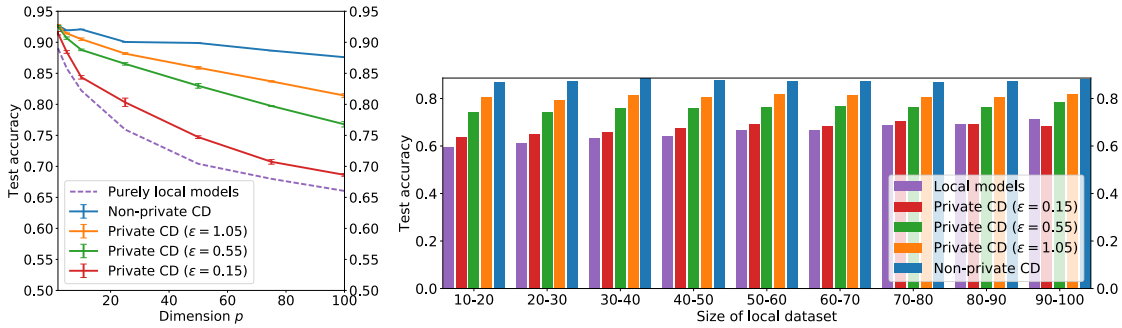
This result shows that the error of the private algorithm after T iterations decomposes into two terms. The first term is the same as in the non-private setting and decreases with T. The second term gives an additive error due to the noise, which takes the form of a weighted sum of the variance of the noise added to the iterate at each iteration (note that we indeed recover the non-private convergence rate of Proposition 2.1 when the noise scale is 0). The number of iterations T thus rules the trade-off between the two terms and should thus be tuned to balance the two sources of error.

In practical scenarios, each user u has an overall privacy budget (ϵ_u, δ_u). Assume that the users agree on a value for T (e.g., using Proposition 2.1 to achieve the desired precision). Each user u is thus expected to wake up $T_u = T/n$ times, and can use Proposition 2.3 to appropriately distribute its privacy budget across the T_u iterations and stop after T_u updates. A simple and practical strategy is to distribute the budget equally across the T_u iterations. Yet, Theorem 2.2 suggests that better utility may be achieved if the noise scale increases with time. We refer to [AB-Conf13] for a characterization of the noise allocation policy which minimizes the utility loss.

Remark 2.7. *Theorem 2.2 implies that a good warm start point $\Theta(0)$ is beneficial. However, $\Theta(0)$ must be obtained in a DP fashion. We show in [AB-Conf13] how to efficiently generate a private warm start in a decentralized manner by propagating perturbed versions of locally trained models in the network, adapting the model propagation approach proposed in [AB-Conf15].*

2.4.3 Differential Privacy for the Full Alternating Procedure

While the previous results give DP guarantees for the model update step, we must also ensure that the graph update step satisfies DP. Interestingly, the latter can essentially be seen as a post-processing step of the model parameters already shared in the model update step. For any $u \in [\![n]\!]$, the only quantity which requires a new query to the local dataset $\mathcal{D}_u$ is the value of the loss function $F_u(\theta_u; \mathcal{D}_u)$ involved in the term $p_{u,\mathcal{V}}$ of the graph update (2.7). As θ_u is fixed during a graph update step, this only needs to be computed once. Under the classic assumption of bounded loss function f, we can



(a) Average test accuracy. (b) Per-user test accuracy w.r.t. local dataset size ($p = 100$).

Figure 2.1: Learning personalized models for linear classification, for different dimensions and several privacy regimes. Results are averaged over 5 runs. Best seen in color.

use a small part of the overall privacy budget of user u to compute a private estimate of $F_u(\theta_u; \mathcal{D}_u)$ using the Laplace or Gaussian mechanism. Finally, to obtain differential privacy guarantees for the full alternating optimization procedure, we apply (simple or advanced) composition over all alternating steps.

2.5 Experiments

In this section, we briefly illustrate the practical behavior of our approach. More results and details can be found in [AB-Conf15]; [AB-Conf13]; [AB-Conf9].

Learning personalized models under known graph. We first show experiments on a linear classification task with $n = 100$ users. Each user has a target linear separator in $\mathbb{R}^p$. The weight between two users u and v is known and given by $w_{u,v} = \exp((\cos(\phi_{u,v}) - 1)/\gamma)$, where $\phi_{u,v}$ is the angle between the target models and $\gamma = 0.1$ (negligible weights are ignored). Each user u receives a random number m_u of training points (between 10 and 100), where each point is drawn uniformly around the origin and labeled according to the target model. We then add some label noise (random flipping with prob. 0.05). We use the ℓ_2-regularized logistic loss (which is 1-Lipschitz), and the regularization parameter of user u is set to $1/m_u > 0$ to ensure overall strong convexity. The hyperparameter λ_1 is tuned to maximize accuracy of the non-private algorithm on a validation set of random problems instances. For each user, the test accuracy of a model is estimated on a separate sample of 100 points. When we enforce privacy constraints, the overall privacy budget is the same for all users, i.e., $\epsilon_u = \epsilon$. A small part of the budget ($\epsilon' = 0.05$) is used to initialize models with a good warm start (see Remark 2.7).

Figure 2.1a shows results obtained with our decentralized CD algorithms for problems of increasing difficulty (by varying the dimension p) with various privacy budgets.

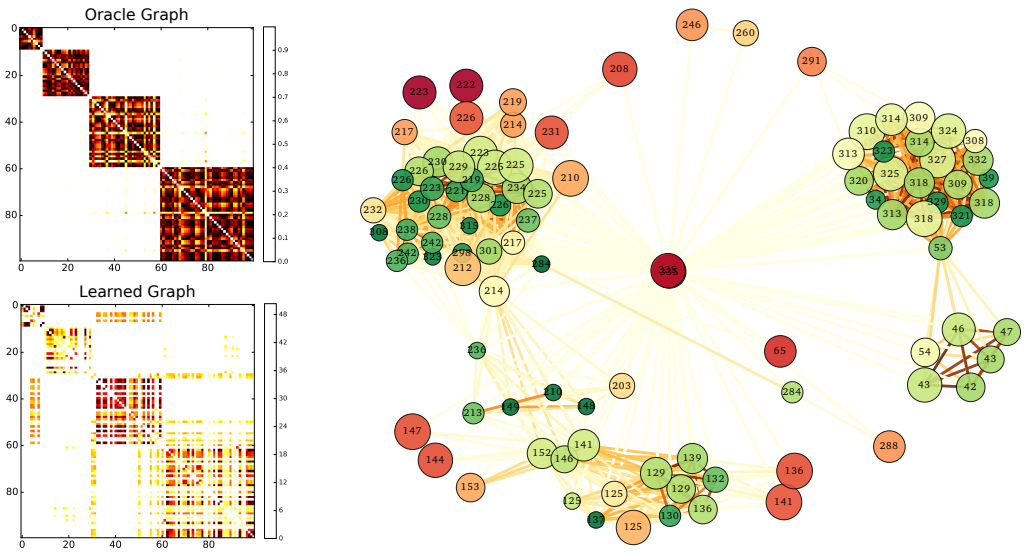


Figure 2.2: Graph learned by our approach on synthetic data. *Left:* Graph weights for the oracle and learned graph (with users grouped by cluster). *Right:* Visualization of the learned graph. The node size is proportional to the user confidence c_u and the color reflects the relative value of the local loss (greener = smaller loss). Nodes are labeled with their rotation angle, and a darker edge color indicates a higher weight. Best seen in color.

The number of iterations per user was tuned based on a validation set of random problem instances. We clearly observe the privacy-utility trade-off, with non-private CD performing best and the test accuracy decreasing with the privacy budget ϵ. However, we see that even in high privacy regimes ($\epsilon = 0.15$) the resulting models significantly outperform the purely local models (a perfectly private baseline), showing the benefits of collaboration. As can be seen in Figure 2.1b, all users (irrespective of their local dataset size) get an improvement in test accuracy. The improvement is especially large for users with small local datasets. By providing more balanced accuracies across users, our approach effectively corrects for the imbalance in dataset size and leads to fair models.

Joint learning of models and graph. We now turn to learning the personalized models and the collaboration graph jointly. For simplicity, we consider a non-private setting. We consider several variants of our algorithms: Perso-Linear learns linear models using our decentralized CD algorithm, and Perso-Boost learns nonlinear models using our decentralized FW algorithm (using decision stumps as base classifiers). Then, in each case, the suffix Learned (resp. Oracle) denotes the variant where the graph is learned (resp. where an oracle graph is given as input). We compare against various competitors, which learn global or personalized models in a centralized or decentralized manner. Global-Boost and Global-Linear learn a single global ℓ_1-Adaboost model (resp. linear

Dataset	Harws	Vehicle Sensor	Computer Buyers	School
Global-Linear	93.64	87.11	62.18	57.06
Local-Linear	92.69	90.38	60.68	70.43
Perso-Linear-Learned	**96.87**	**91.45**	69.10	***71.78***
Global-Boost	94.34	88.02	***69.16***	69.96
Local-Boost	93.16	90.59	66.61	70.69
Perso-Boost-Learned	***95.57***	***91.04***	**73.55**	**72.47**

Table 2.1: Test accuracy (%) on real datasets, averaged over 3 runs. Best results in boldface, second best in italic.

model) over the centralized dataset $\mathcal{D} = \cup_u \mathcal{D}_u$. Local-Boost and Local-Linear learn purely local personalized models independently for each user. Models are initialized to zero vectors and the initial graphs of Perso-Linear-Learned and Perso-Boost-Learned are learned using the purely local models, and then updated after every 100 iterations of optimizing the models, with $\kappa = 5$.

We first illustrate the behavior of Perso-Boost on a synthetic problem constructed from the classic two interleaving Moons dataset which has nonlinear class boundaries. We consider $n = 100$ users, clustered in 4 groups of 10, 20, 30 and 40 users. Users in the same cluster are associated with a similar rotation of the feature space and hence have similar tasks. To assess the effectiveness of our graph learning procedure, we construct an "oracle" collaboration graph based on the difference in rotation angles between users. Each user u obtains a training sample whose size m_u is drawn randomly between 3 and 15. The data dimension is 20 and the number of base predictors is $K = 200$. Perso-Boost-Learned outperforms all other approaches on this problem, except Perso-Boost-Oracle for it makes use of the oracle graph computed from the true data distributions. Figure 2.2 (left) shows that the graph learned by Perso-Boost-Learned is able to approximately recover the ground-truth cluster structure. Figure 2.2 (right) provides a more detailed visualization of the learned graph, which shows the effect of the inductive bias brought by the confidence-weighted loss term in our objective function (2.1) discussed in Section 2.1. Indeed, users with high confidence and high loss values tend to have small degrees while users with low confidence or low loss values are more densely connected.

Finally, we present results on real datasets that are naturally collected at the user level: Human Activity Recognition With Smartphones (Harws, $n = 30$), Vehicle Sensor ($n = 23$), Computer Buyers ($n = 190$) and School ($n = 140$).[6] As shown in Table 2.1, our methods Perso-Boost-Learned and Perso-Linear-Learned achieve the best performance. This demonstrates the relevance and usefulness of our approach when dealing with heterogeneous local datasets, even when no prior information is available to build a predefined collaboration graph.

[6] Anguita et al. (2013), Duarte and Hu (2004), Goldstein (1991), https://github.com/probml/pmtkdata/.

Chapter 3

Decentralized and Private Learning with Pairwise Loss Functions

Existing decentralized learning algorithms designed for the empirical risk minimization framework, including those presented in Chapter 2, assume that the data-dependent term in the objective function can be written as a *sum of local objectives* $\sum_{u=1}^{n} F_u(\theta_u; \mathcal{D}_u)$, see (1.2) and (2.1). This separable structure plays a key role in the design of decentralized algorithms, as it implies that each user u can locally evaluate her part of the global objective and compute gradients of F_u with respect to the model parameters. This naturally leads to the classic structure of decentralized algorithms, which is to alternate between local update steps performed independently by each user, and communication steps to aggregate and synchronize their local parameters so as to ensure convergence to an optimum (or stationary point) of the global objective **[AB-Journal2]**. While the separability assumption allows to efficiently tackle a large class of ML problems in a decentralized manner, many learning objectives do not fit into this framework.

In this chapter, we are interested in the design of decentralized learning algorithms that can solve problems that involve a *pairwise loss function* $h : \mathcal{C} \times \mathcal{X} \times \mathcal{X} \to \mathbb{R}$:

$$\arg\min_{\theta \in \mathcal{C}} \Big\{ H(\theta; \mathcal{D}) = \frac{1}{m^2} \sum_{u,v=1}^{n} \sum_{x \in \mathcal{D}_u, x' \in \mathcal{D}_v} h(\theta; x, x') \Big\}. \tag{3.1}$$

Many machine learning problems can be cast as (3.1), see (Kar et al., 2013) **[AB-Journal9]**. Notable examples include bipartite ranking (Clémençon et al., 2008) **[AB-Conf2]**, metric learning (Kulis, 2013) **[AB-Book1]**, clustering (Clémençon, 2014) and graph inference (Biau and Bleakley, 2006) **[AB-Conf18]**. Unfortunately, the objective in (3.1) does not decompose into a sum of local objectives, since most of the terms involve pairs of data points coming from different local datasets. For such problems, some form of data exchange appears to be necessary, and this may be at odds with the objectives of decentralized and privacy-preserving machine learning.

Setting aside privacy issues for the moment, our first contribution to propose *efficient decentralized methods to estimate and optimize pairwise objectives* of the form (3.1). Our algorithms belong to the family of randomized gossip protocols (Kempe et al., 2003; Boyd et al., 2006), which are lightweight decentralized algorithms where each user communicates with a randomly chosen neighbor in the network graph $\mathcal{G}$ at each step.[1] Our methods combine two types of iterative information exchange: a classic averaging step to aggregate local estimates, and a *novel data propagation step* which allows each user to access data points from other users. We first show how to design synchronous and asynchronous gossip algorithms for decentralized pairwise estimation, where the goal is to evaluate a pairwise objective of the form (3.1) for fixed θ. This is an interesting problem in its own right as it corresponds to estimating U-statistics (Lee, 1990), a general class of estimates which include popular quantities such as the sample variance, Gini mean difference, Kendall's tau coefficient, Wilcoxon Mann-Whitney hypothesis test (Mann and Whitney, 1947), Rényi-2 entropy (Acharya et al., 2015) and Area under the ROC Curve (AUC) (Bradley, 1997). We then combine our data propagation step with ideas from dual averaging optimization (Nesterov, 2009) to design gossip protocols to solve (3.1) in the convex setting. We formally prove the convergence of our algorithms, and show that our analysis allows to capture some degree of data and network-dependence in the rates.

The above algorithms require users to share individual data points, which raises privacy issues when data is sensitive. To address such privacy constraints, we consider the local model of differential privacy in which users share only perturbed versions of their data. We propose a generic locally private protocol to estimate pairwise statistics, which can be readily used to obtain private versions of our gossip algorithms. We also propose a specific protocol to compute the AUC based on hierarchical histograms, and a protocol which operates in a slightly relaxed model where pairs of users can rely on 2-party secure computation to obtain better utility. The practical behavior of all our approaches is illustrated on a series of experiments on pairwise estimation and learning tasks.

This chapter provides a unified treatment of the work published in a series of three conferences papers [**AB-Conf21**]; [**AB-Conf16**]; [**AB-Conf3**].

Related work. In the centralized setting, empirical risk minimization with pairwise losses has been extensively studied as it raises specific statistical and algorithmic challenges. Clémençon et al. (2008) [**AB-Journal9**] provided statistical learning guarantees, Kar et al. (2013) and Boissier et al. (2016) studied the online learning setting, and [**AB-Conf23**] designed efficient SGD-based algorithms. Recent work also studied how to enforce differential privacy in the trusted curator model (Huai et al., 2020; Yang et al., 2021).

In decentralized learning, existing work has focused on the estimation and optimization of objectives that are separable across users, see (Lian et al., 2017; Tang et al., 2018;

[1] Note that in contrast to the approach presented in Chapter 2, here the network graph $\mathcal{G}$ is fixed and simply encodes communication constraints (i.e., it does not carry any semantic information).

Assran et al., 2019; Scaman et al., 2019; Koloskova et al., 2020) for recent work and **[AB-Journal2]** (Section 2.1 therein) for an overview. More specifically, gossip protocols have been proposed for computing averages, sums, quantiles and other separable aggregate statistics (Kempe et al., 2003; Boyd et al., 2006; Mosk-Aoyama and Shah, 2008), see Shah (2009) and Dimakis et al. (2010) for surveys. Gossip algorithms have then been proposed to optimize sums of local objectives (see Nedic and Ozdaglar, 2009; Ram et al., 2010; Duchi et al., 2012; Wei and Ozdaglar, 2012; Tsianos et al., 2015; Lian et al., 2018; Koloskova et al., 2019; Hendrikx et al., 2019, among others). Our work **[AB-Conf16]** was the first one to propose decentralized algorithms for pairwise optimization. In **[AB-Conf11]**, we proposed scalable distributed pairwise optimization algorithms for cluster computing, where communication constraints and efficiency trade-offs are different.

To the best of our knowledge, our work **[AB-Conf3]** is the first to propose protocols to estimate pairwise statistics in the local model of DP. Most previous work has focused on computing quantities that are separable across individual users, such as sums and histograms (see e.g., Bassily and Smith, 2015; Wang et al., 2017; Kulkarni et al., 2019; Cormode et al., 2018; Bassily et al., 2017). These existing protocols cannot be directly used to compute U-statistics due to the pairwise nature of the terms. Still, we are able to re-use some ideas, in particular the use of private hierarchical histograms.

Outline of the chapter. Section 3.1 presents our gossip algorithms for decentralized pairwise estimation, before moving to pairwise optimization in Section 3.2. Section 3.3 introduces locally private algorithms for pairwise estimation. Finally, Section 3.4 presents some numerical experiments.

Remark 3.1. *For simplicity of presentation, throughout this chapter we will focus on the case where each user holds a single data point (representing its personal data), i.e.,* $\mathcal{D} = \{x_1, \dots, x_n\}$ *with* $\mathcal{D}_u = \{x_u\}$ *for each user* $u \in [\![n]\!]$*. Our methods naturally extend to the general case.*

3.1 Gossip Algorithms for Pairwise Estimation

In this section, we focus on the problem of *decentralized pairwise estimation* without privacy constraint. Let $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a measurable function, symmetric in its two arguments and with $h(x, x) = 0$, $\forall x \in \mathcal{X}$. Let $x_1, \dots, x_n \in \mathcal{X}$ where x_u is the data point held by user $u \in [\![n]\!]$. The goal is to compute the following quantity, known as a degree two U-statistic with kernel h (Hoeffding, 1948; Lee, 1990):[2]

$$U_{h,n} = \frac{1}{n^2} \sum_{u,v=1}^{n} h(x_u, x_v). \tag{3.2}$$

[2]The usual definition of U-statistic is the average over the $n(n-1)/2$ pairs of *distinct* points. For convenience, in this section we consider the sum over all n^2 pairs, which is equivalent up to a factor of $(n-1)/2n$.

Algorithm 3.1 GoSta-sync: synchronous gossip algorithm for computing (3.2).

Input: Network graph $\mathcal{G} = ([\![n]\!], E)$; Data point x_u for each user $u \in [\![n]\!]$

1: Each user $u \in [\![n]\!]$ initializes its auxiliary observation $a_u = x_u$ and its estimate $z_u = 0$
2: **for** $t = 1, 2, \ldots$ **do**
3: **for** $u = 1, \ldots, n$ **do**
4: Set $z_u \leftarrow \frac{t-1}{t} z_u + \frac{1}{t} h(x_u, a_u)$
5: Draw $\{u, v\}$ uniformly at random from E
6: Set $z_u, z_v \leftarrow \frac{1}{2}(z_u + z_v)$
7: Swap auxiliary observations of users u and v: $a_u \leftrightarrow a_v$
8: **return** Each user u has z_u

U-statistics are commonly used as point estimators of various global properties of distributions and in statistical hypothesis testing (Lee, 1990; Mann and Whitney, 1947). For instance, the Gini mean difference, a classic measure of dispersion, corresponds to the case where $\mathcal{X} \subset \mathbb{R}$ and $h(x, x') = |x - x'|$. Note that computing (3.2) can be seen as evaluating the objective function in (3.1) at a fixed θ,[3] and is thus a natural stepping stone towards solving (3.1).

In this section, we propose and analyze algorithms to efficiently compute (3.2) in a decentralized manner. As explained in Chapter 1, we consider a fixed network topology encoded as a connected undirected graph $\mathcal{G} = ([\![n]\!], E)$, where nodes correspond to users and the presence of an edge $\{u, v\} \in E$ means that users u and v can exchange messages. Our algorithms belong to the family of (randomized) gossip protocols (Boyd et al., 2006), where each user exchanges information with a single (random) neighbor at a time.

Algorithm. For simplicity, we focus here on a *synchronous* setting: users have access to a global clock with counter t so that they can all update their local estimate at each time instance. We stress the fact that the users need not be aware of the global network topology as they will only interact with their direct neighbors in the graph $\mathcal{G}$.

Our approach is based on the observation that $U_{h,n} = \frac{1}{n} \sum_{u=1}^{n} \overline{h}_u$, with $\overline{h}_u = \frac{1}{n} \sum_{v=1}^{n} h(x_u, x_v)$, and we write $\overline{h} = (\overline{h}_1, \ldots, \overline{h}_n)^\top$. Computing $U_{h,n}$ is thus similar to the classic decentralized averaging problem (Boyd et al., 2006), with the key difference that each "local" value $\overline{h}_u$ is itself an average which depends on the entire data sample $\mathcal{D}$. Consequently, our algorithm combines two steps at each iteration: a *data propagation step* to allow each user u to estimate $\overline{h}_u$, and an *averaging step* to ensure convergence to the desired value $U_{h,n}$.

Formally, let us denote by z_u the (local) estimate of $U_{h,n}$ by user u at a given time step. In order to propagate data across the network, each user u maintains an auxiliary observation a_u, initialized to x_u. At each time step, each user u updates its local estimate by taking the running average of z_u and $h(x_u, a_u)$. Then, an edge $\{u, v\} \in E$ of the

[3] It can also be seen as a special case of (3.1) for $h(\theta; x, x') = (\theta - h(x, x'))^2$.

network is drawn uniformly at random, and the corresponding pair of users u and v average their local estimates z_u and z_v and swap their auxiliary observations a_u and a_v. Each data point is thus performing a (coupled) random walk on the network graph. The full algorithm, coined GoSta, is given in Algorithm 3.1.

Convergence analysis. We prove that the local estimates generated by Algorithm 3.1 converge to $U_{h,n}$ and precisely characterize the convergence rate.

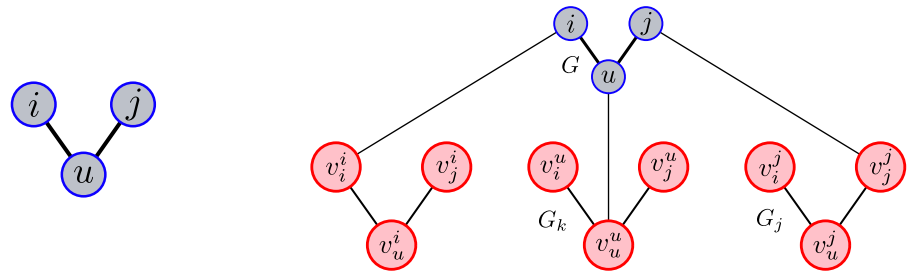
Theorem 3.1 ([AB-Conf21]). *Let $\mathcal{G}$ be connected and non-bipartite, and denote by $z(t) = (z_1(t), \dots, z_n(t))^\top \in \mathbb{R}^n$ the vector of local estimates generated by Algorithm 3.1 at iteration $t \in \mathbb{N}$. For all $u \in [\![n]\!]$, we have $\lim_{t\to+\infty} \mathbb{E}[z_u(t)] = U_{h,n}$. Moreover, for any $t > 0$,*

$$\|\mathbb{E}[z(t)] - U_{h,n}\mathbb{1}_n\|_2 \leq \frac{1}{c_{\mathcal{G}} t}\left\|\overline{h} - U_{h,n}\mathbb{1}_n\right\|_2 + \left(\frac{2}{c_{\mathcal{G}} t} + e^{-c_{\mathcal{G}} t}\right)\left\|H - \overline{h}\mathbb{1}_n^\top\right\|_2, \tag{3.3}$$

where $\mathbb{1}_n = (1, ..., 1)^\top \in \mathbb{R}^n$, $H = [h(x_u, x_v)]_{1\leq u,v\leq n} \in \mathbb{R}^{n\times n}$, and $c_{\mathcal{G}} = \beta_{\mathcal{G}}/|E| > 0$ where $\beta_{\mathcal{G}}$ is the spectral gap of the graph $\mathcal{G}$.

Sketch of proof. To analyze Algorithm 3.1, we consider an equivalent reformulation which allows us to model the data propagation and averaging steps separately. Specifically, for each $u \in [\![n]\!]$, we define a "phantom" $\mathcal{G}_u = (V_u, E_u)$ of the original graph $\mathcal{G}$, with $V_u = \{v_i^u : 1 \leq i \leq n\}$ and $E_u = \{(v_i^u, v_j^u) : (i,j) \in E\}$. We then create a new graph $\tilde{\mathcal{G}} = (\tilde{V}, \tilde{E})$ where each node $u \in [\![n]\!]$ is connected to its counterpart $v_u^u \in V_u$, as illustrated in Figure 3.1. In this new graph, the nodes $1, \dots, n$ from the original graph will hold the estimates $z_1(t), \dots, z_n(t)$, while the role of each $\mathcal{G}_u$ is to simulate data propagation in the original graph $\mathcal{G}$. For $i \in [\![n]\!]$, $v_i^u \in V^u$ initially holds the value $h(x_u, x_i)$. At each iteration of Algorithm 3.1, we draw a random edge $\{i,j\}$ of $\mathcal{G}$ and nodes v_i^u and v_j^u swap their value for all $u \in [\![n]\!]$. To update its estimate, each node u will use the current value at v_u^u. This construction allows us to represent the system state at iteration t by a vector $S(t) = (S_1(t)^\top, S_2(t)^\top)^\top \in \mathbb{R}^{n+n^2}$. The first n coefficients, $S_1(t)$, are associated with nodes in $[\![n]\!]$ and correspond to the estimate vector $z(t) = [z_1(t), \dots, z_n(t)]^\top$. The last n^2 coefficients, $S_2(t)$, are associated with nodes in $(V_u)_{1\leq u\leq n}$ and represent the data propagation in the network. At a given step $t > 0$, we are interested in the transition matrix $M(t)$ such that $\mathbb{E}[S(t+1)] = M(t)\mathbb{E}[S(t)]$. The transition matrix $M(t)$ accounts for three events: the *averaging step* (the action of $\mathcal{G}$ on itself), the *data propagation* (the action of $\mathcal{G}_u$ on itself for all $u \in [\![n]\!]$) and the *estimate update* (the action of $\mathcal{G}_u$ on node u for all $u \in [\![n]\!]$). Each event corresponds to a block in $M(t)$. We precisely characterize the matrix $M(t)$ and the convergence result is then obtained by analyzing its spectrum, similarly to standard gossip averaging (Boyd et al., 2006). □

Theorem 3.1 shows that the local estimates generated by Algorithm 3.1 converge to $U_{h,n}$ at a rate of $O(1/t)$. Furthermore, the constants in (3.3) reveal the rate dependency



(a) Original graph $\mathcal{G}$. (b) Phantom graph $\tilde{\mathcal{G}}$.

Figure 3.1: Illustration of the "phantom graph" construction used in our analysis.

on the particular problem instance. Indeed, the two norm terms are *data-dependent* and quantify the difficulty of the estimation problem itself through a measure of dispersion of the values of the kernel h across users. Assuming that h is Lipschitz, this captures the impact of a certain form of data heterogeneity (the variance of data across users). On the other hand, $c_{\mathcal{G}}$ is *network-dependent*. The spectral gap $\beta_{\mathcal{G}}$ is a measure of the diffusion speed in the graph $\mathcal{G}$, and graphs with a larger spectral gap typically have better connectivity (Chung, 1997). This will be illustrated in the experiments of Section 3.4.

Remark 3.2 (Extension to the asynchronous setting)**.** *We also considered the asynchronous setting, where there is no global clock to synchronize updates. Instead, each user has an independent local Poisson clock and wakes up when it ticks. At any time step, a random user u wakes up and exchanges information with a random neighbor v. In [**AB-Conf21**], we proposed an asynchronous variant of our algorithm in which u and v use unbiased estimates of the global iteration number to update z_u and z_v, and proved that this algorithm converges at a rate of $O(\log t/t)$.*

3.2 Gossip Algorithms for Pairwise Optimization

We now turn to the problem of *decentralized pairwise optimization*, still setting aside privacy concerns for now. Let $\theta \in \mathbb{R}^p$ be the vector of model parameters, and let $h : \mathbb{R}^p \times \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a pairwise loss function which we assume to be differentiable and convex in its first argument. We further assume that for any $(x, x') \in \mathcal{X}^2$, there exists $L_h > 0$ such that $h(\cdot; x, x')$ is L_h-Lipschitz (with respect to the ℓ_2-norm). Let $\psi : \mathbb{R}^p \to \mathbb{R}^+$ be a non-negative, convex, possibly non-smooth regularizer with $\psi(0) = 0$ for simplicity. We aim at solving the following general optimization problem, which is equivalent to (3.1) with an explicit (but optional) regularization term:

$$\arg\min_{\theta \in \mathbb{R}^p} \Big\{ H(\theta; \mathcal{D}) = \frac{1}{n^2} \sum_{1 \le u, v \le n} h(\theta; x_u, x_v) + \psi(\theta) \Big\}. \tag{3.4}$$

Such pairwise objectives appear as empirical risk measures in many machine learning problems (Kar et al., 2013) [**AB-Journal9**]. For instance, in bipartite ranking (Clémençon

et al., 2008) **[AB-Conf2]** and imbalanced classification (Herschtal and Raskutti, 2004; Zhao et al., 2011), a popular objective is to maximize the Area under the ROC Curve (AUC). Given binary labels $y_1, \ldots, y_n \in \{-1, 1\}$ associated with each data point, the goal is to learn a scoring rule $x \mapsto s_\theta(x)$ which gives larger scores to positive data points than to negative ones. This can be done by resorting to a convex surrogate of the AUC such as

$$h(\theta; x_u, x_v) = \mathbb{I}[y_u > y_v] \log\left(1 + e^{s_\theta(x_v) - s_\theta(x_u)}\right), \tag{3.5}$$

where $\mathbb{I}[\cdot]$ is the indicator function. The regularization term $\psi(\theta)$ can be the ℓ_2-norm of θ, or non-smooth norms (e.g., ℓ_1-norm) when a sparse model is desired (Bach et al., 2012).

For notational convenience, we denote $h_u(\cdot) = (1/n) \sum_{v=1}^n h(\cdot; x_u, x_v)$ for $u \in [\![n]\!]$ and $\bar{h}^n(\cdot) = (1/n) \sum_{u=1}^n h_u(\cdot)$. We can thus rewrite the objective as $H(\theta; \mathcal{D}) = \bar{h}^n(\theta) + \psi(\theta)$. Note that the function $\bar{h}^n$ is L_h-Lipschitz, since all the h_u are L_h-Lipschitz.

In the following, we will combine some ideas from our decentralized pairwise estimation algorithm of Section 3.1 (in particular the data propagation step) with dual averaging techniques from convex optimization (Nesterov, 2009) to obtain the first decentralized algorithms for solving (3.4). Before presenting our approach, we start with a quick reminder on centralized dual averaging.

3.2.1 Reminder on Centralized Dual Averaging

In this section, we review the stochastic dual averaging optimization algorithm (Nesterov, 2009; Xiao, 2010) to solve Problem (3.4) in the centralized setting. To explain the main idea behind dual averaging, let us first consider the iterations of Stochastic Gradient Descent (SGD), assuming $\psi \equiv 0$ for simplicity:

$$\theta(t+1) = \theta(t) - \gamma(t) g(t),$$

where $\mathbb{E}[g(t)|\theta(t)] = \nabla \bar{h}^n(\theta(t))$, and $(\gamma(t))_{t \geq 0}$ is a non-negative non-increasing step size sequence. For SGD to converge to an optimal solution, the step size sequence must satisfy $\gamma(t) \underset{t \to +\infty}{\longrightarrow} 0$ and $\sum_{t=0}^\infty \gamma(t) = \infty$. As noticed by Nesterov (2009), an undesirable consequence is that new gradient estimates are given smaller weights than old ones. Dual averaging aims at integrating all gradient estimates with the same weight.

Let $(\gamma(t))_{t \geq 0}$ be a positive and non-increasing step size sequence. The dual averaging algorithm maintains a sequence of iterates $(\theta(t))_{t > 0}$, and a sequence $(z(t))_{t \geq 0}$ of "dual" variables which collect the sum of the unbiased gradient estimates seen up to time t. The algorithm starts with $\theta(1) = z(0) = 0$. Then, at each step $t > 0$, it computes an unbiased estimate $g(t)$ of the gradient of the differentiable part of the objective, in our case $\nabla \bar{h}^n(\theta(t))$. The most common choice is to take $g(t) = \nabla h(\theta(t); x_{u_t}, x_{v_t})$ where u_t and v_t are drawn uniformly at random from $[\![n]\!]$. One then sets $z(t+1) = z(t) + g(t)$ and

generates the next iterate with the following rule:

$$\theta(t+1) = \pi_t^{\psi}(z(t+1)), \quad \text{with } \pi_t^{\psi}(z) = \arg\min_{\theta \in \mathbb{R}^d} \left\{ -z^\top \theta + \frac{\|\theta\|^2}{2\gamma(t)} + t\psi(\theta) \right\}.$$

When clear from the context, we will drop the dependence in ψ and simply write $\pi_t(z)$.

Remark 3.3. *Note that π_t is related to the* proximal operator *of a function $\phi : \mathbb{R}^d \to \mathbb{R}$ defined by $\operatorname{prox}_\phi(x) = \arg\min_{z \in \mathbb{R}^d} \left(\|z - x\|^2/2 + \phi(x) \right)$. Indeed, one can write $\pi_t(z) = \operatorname{prox}_{t\gamma(t)\psi}(\gamma(t)z)$. For many functions ψ of practical interest, π_t has a closed form solution (Parikh and Boyd, 2013). For instance, when $\psi = \|\cdot\|^2$, π_t is a simple scaling, and when $\psi = \|\cdot\|_1$ it is a soft-thresholding operator. If ψ is the indicator function of a closed convex set $\mathcal{C}$, then π_t is the projection operator onto $\mathcal{C}$.*

3.2.2 Proposed Approach

Our approach relies on the stochastic dual averaging method presented above. This choice is guided by the fact that the structure of the updates makes dual averaging easier to analyze in the decentralized setting than (sub)gradient descent when the problem is constrained or regularized. Indeed, dual averaging maintains a simple sum of gradients, while the (non-linear) smoothing operator π_t is applied separately.

More precisely, our work builds upon the work of Duchi et al. (2012), who proposed a distributed dual averaging algorithm to optimize an average of *univariate* functions $f(\cdot; x_u)$ as in (1.1). In their algorithm, each user u computes *unbiased* estimates of $\nabla f(\cdot; x_u)$ that are iteratively averaged over the network. Unfortunately, in our setting, user u cannot compute unbiased estimates of $\nabla h_u(\cdot) = \nabla(1/n)\sum_{v=1}^n h(\cdot; x_u, x_v)$ as the latter depends on all data points. To go around this problem, we rely on the gossip data propagation step introduced for decentralized estimation (see Section 3.1) so that users can compute *biased* estimates of $\nabla h_u(\cdot)$ while keeping small communication and memory overhead.

Algorithm. We consider here the synchronous setting where each user has access to a global clock and every user execute updates simultaneously at each tick of the clock. We assume that the step size sequence $(\gamma(t))_{t \geq 0}$ is the same for every user. At any time, each user u holds the following quantities in its local memory: a gradient accumulator z_u, its original observation x_u, and an auxiliary observation a_u, which is initialized at x_u but will change throughout the algorithm as a result of data propagation.

Our randomized gossip algorithm, given in Algorithm 3.2, goes as follows. At each time step, an edge $\{u, v\} \in E$ is drawn uniformly at random. Then, users u and v average their gradient accumulators z_u and z_v, and swap their auxiliary observations a_u and a_v. Finally, every user of the network performs a dual averaging step, using their original observation and their current auxiliary one to estimate the partial gradient.

Algorithm 3.2 Synchronous gossip dual averaging for solving (3.4).

Input: Network graph $\mathcal{G} = ([\![n]\!], E)$; Data point x_u for each user $u \in [\![n]\!]$; Step sizes $(\gamma(t))_{t\geq 1}$.

Each node u initializes $a_u = x_u$, $z_u = \theta_u = \bar{\theta}_u = 0$.
for $t = 1, \ldots, T$ **do**
  Draw $\{u, v\}$ uniformly at random from E
  Set $z_u, z_v \leftarrow \frac{z_u + z_v}{2}$
  Swap auxiliary observations: $a_u \leftrightarrow a_v$
  for $u = 1, \ldots, n$ **do**
    Update $z_u \leftarrow z_u + \nabla_\theta h(\theta_u; x_u, a_u)$
    Compute $\theta_u \leftarrow \pi_t(z_u)$
    Average $\bar{\theta}_u \leftarrow \left(1 - \frac{1}{t}\right)\bar{\theta}_u + \frac{1}{t}\theta_u$
return Each user u has $\bar{\theta}_u$

Convergence analysis. The following proposition adapts the convergence rate of centralized dual averaging to our decentralized algorithm.

Theorem 3.2 ([AB-Conf16]). *Let $\mathcal{G}$ be a connected and non-bipartite graph with n nodes, and let $\theta^* \in \arg\min_{\theta\in\mathbb{R}^d} H(\theta; \mathcal{D})$. Let $(\gamma(t))_{t\geq 1}$ be a non-increasing and non-negative sequence. For any $u \in [\![n]\!]$ and any $t \geq 0$, let $z_u(t) \in \mathbb{R}^d$, $\bar{\theta}_u(t) \in \mathbb{R}^d$ and $a_u(t) \in \mathcal{X}$ be generated according to Algorithm 3.2. Then for any $u \in [\![n]\!]$ and $T > 1$, we have:*

$$\mathbb{E}[H(\bar{\theta}_u; \mathcal{D}) - H(\theta^*; \mathcal{D})] \leq C_1(T) + C_2(T) + C_3(T), \quad \textit{where}$$

$$C_1(T) = \frac{1}{2T\gamma(T)}\|\theta^*\|^2 + \frac{L_h^2}{2T}\sum_{t=1}^{T-1}\gamma(t), \quad C_2(T) = \frac{3L_h^2}{T\left(1 - \sqrt{1 - \beta_{\mathcal{G}}/|E|}\right)}\sum_{t=1}^{T-1}\gamma(t),$$

$$C_3(T) = \frac{1}{T}\sum_{t=1}^{T-1}\mathbb{E}_t[(\pi_t(\bar{z}^n(t)) - \theta^*)^\top \bar{\epsilon}^n(t)],$$

with $\bar{\epsilon}^n(t) = \frac{1}{n}\sum_{u=1}^n \left(\nabla h(\theta_u(t); x_u, a_u(t)) - \frac{1}{n}\sum_{v=1}^n \nabla h(\theta_u(t); x_u, x_v)\right)$ the average gradient bias at time t, and $\beta_{\mathcal{G}}$ the spectral gap of the graph $\mathcal{G}$.

The rate of convergence in Theorem 3.2 is divided into three parts: $C_1(T)$ is a *data dependent* term which corresponds to the rate of convergence of centralized dual averaging, while $C_2(T)$ and $C_3(T)$ are *network dependent* terms. Indeed, $C_2(T)$ depends on the spectral gap $\beta_{\mathcal{G}}$, similarly to the case of decentralized estimation (Theorem 3.1). The convergence rate of our algorithm thus improves when the spectral gap is large, which is typically the case for well-connected graphs (Chung, 1997). Setting $\gamma(t) \propto 1/\sqrt{t}$ ensures that $C_1(T)$ and $C_2(T)$ converge to 0 at a rate of $O(1/\sqrt{T})$. Finally, the term $C_3(T)$ comes from the bias of our partial gradient estimates. In practice, $C_3(T)$ vanishes quickly and has a small impact on the rate of convergence, as shown in Section 3.4. For a formal analysis of this bias term, see (Colin, 2016).

Remark 3.4 (Extension to the asynchronous setting)**.** *We also proposed and analyzed a variant of our algorithm for the asynchronous setting. This is especially challenging because there is a need for a common time scale to perform a suitable decrease in the step size $\gamma(t)$. We use unbiased estimates of t to maintain consistent step sizes across the network, and introduce appropriate weights for the gradient update and model averaging steps to ensure that all users asymptotically count equally in every gradient accumulator. Details can be found in [**AB-Conf16**].*

3.3 Locally Private Protocols for Pairwise Estimation

The approaches presented so far for decentralized estimation and optimization did not consider any privacy constraint. In fact, they are based on propagating data across users, which is not possible when data is sensitive. In this section, we propose an approach for privacy-preserving pairwise estimation in the local model of differential privacy, in which each user randomizes its data point locally before sharing it.

3.3.1 Problem Setting

Statistical framework. We will consider the following statistical framework. Let ω be an (unknown) distribution over an input space $\mathcal{X}$ and $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a pairwise function (assumed to be symmetric for simplicity). Given a sample $\mathcal{D} = \{x_u\}_{u=1}^n$ of n data points drawn from ω, we are interested in estimating the following quantity:

$$U_h = \mathbb{E}_{X_1, X_2 \sim \omega}[h(X_1, X_2)]. \tag{3.6}$$

The unbiased estimate of (3.6) with minimum variance is the U-statistic of degree 2 (Hoeffding, 1948; Lee, 1990) with kernel h:

$$U_{h,n} = \frac{2}{n(n-1)} \sum_{1 \le u < v \le n} h(x_u, x_v). \tag{3.7}$$

Up to a constant scaling factor, (3.7) is equivalent to the quantity (3.2) considered in our decentralized pairwise estimation of Section 3.1, hence with a slight abuse of notation we also denote it by $U_{h,n}$. As discussed in Section 3.1, U-statistics include many statistics of interest, among which the sample variance, Gini mean difference, Kendall's tau coefficient, Wilcoxon Mann-Whitney hypothesis test and Area under the ROC Curve (AUC).

Local Differential Privacy (LDP). We consider the *local* model of DP (Kasiviswanathan et al., 2008; Duchi et al., 2013), which captures the setting where individuals do not trust anyone and use a local randomizer to perturb their input data before sharing it. This is formalized by the following definition, which is equivalent to standard DP (Definition 1.1) in the case where datasets have size 1.

Algorithm 3.3 LDP protocol to estimate (3.7) using quantization and private histograms.

Public parameters: Privacy budget ϵ, number of bins k, quantization scheme $Q : \mathcal{X} \to [\![k]\!]$
Input: Data point $x_u \in \mathcal{X}$ for each user $u \in [\![n]\!]$
1: **for** each user $u \in [\![n]\!]$ **do**
2: Form quantized input $Q(x_u) \in [\![k]\!]$
3: For $\beta = k/(k + e^{\epsilon} - 1)$, generate $\tilde{x}_u \in [\![k]\!]$ s.t.

$$P(\tilde{x}_u = i) = \begin{cases} 1 - \beta & \text{for } i = Q(x_u), \\ \beta/k & \text{for } i \neq Q(x_u), \end{cases} \tag{3.8}$$

4: Send $\tilde{x}_u$ to the untrusted aggregator
5: **return** $\widehat{U}_{h,n}$ computed from $\tilde{x}_1, \ldots, \tilde{x}_n$ and β as in (3.9)

Definition 3.1 (Duchi et al., 2013)**.** *Let $\mathcal{R}$ be a local randomizer which takes a single data point as input. Given $\epsilon, \delta > 0$, we say that $\mathcal{R}$ is (ϵ, δ)-locally differentially private, or (ϵ, δ)-LDP, if for all pairs of data points $x, x' \in \mathcal{X}$ and for all $\mathcal{O} \subseteq \mathrm{range}(\mathcal{R})$, we have:*

$$\Pr[\mathcal{R}(x) \in \mathcal{O}] \leq e^{\epsilon} \Pr[\mathcal{R}(x') \in \mathcal{O}] + \delta.$$

Throughout this section we assume the presence of a (honest) *untrusted aggregator* who collects the randomized inputs $\mathcal{R}(x_1), \ldots, \mathcal{R}(x_n)$ and processes them to compute an estimate of (3.6). Nevertheless, one of the protocols we will introduce can be directly used to obtain private versions of the decentralized estimation and optimization algorithms introduced in Section 3.1 and Section 3.2 (see Remark 3.5).

Note that due to the pairwise nature of the terms in (3.7), accurate LDP protocols for computing U-statistics cannot be straightforwardly obtained by resorting to existing protocols. Indeed, one cannot apply the local randomizer to the terms of the sum based on the sensitivity of h (as each term is shared across two users), and perturbing the inputs can lead to large errors when passed through the (potentially discontinuous) function h.

3.3.2 Generic Locally Private Protocol from Quantization

Discrete inputs. We first consider the case of discrete inputs taking one of k values. The possible values of the kernel function can be written as a matrix $A \in \mathbb{R}^{k \times k}$ where $A_{ij} = h(i, j)$. In this case, we let the local randomizer $\mathcal{R}$ to be k-ary randomized response (Kairouz et al., 2014): the perturbed input $\mathcal{R}(x_u)$ is set to the true input x_u with some probability $1 - \beta$ and a uniformly random value with probability β, as shown in (3.8). Let e_i denote the vector of length k with a one in the i-th position and 0 elsewhere. For each perturbed input in one-hot encoding form $e_{\mathcal{R}(x_u)}$ we can deduce an unbiased estimate of e_{x_u}. As the discrete U-statistic is a linear function of each of these vectors, computing it

on these unbiased estimates gives an unbiased estimate $\widehat{U}_{h,n}$ which can be written as:

$$\widehat{U}_{h,n} = \frac{1}{\binom{n}{2}} \sum_{1 \le u < v \le n} \widehat{h}_A(\mathcal{R}(x_u), \mathcal{R}(x_v)), \tag{3.9}$$

and is itself a U-statistic with kernel $\widehat{h}_A$ given by

$$\widehat{h}_A(\mathcal{R}(x_1), \mathcal{R}(x_2)) = (1-\beta)^{-2}(e_{\mathcal{R}(x_1)} - b)^\top A(e_{\mathcal{R}(x_2)} - b), \tag{3.10}$$

where b is the vector of length k with every entry β/k. Details and analysis of this process, leveraging Hoeffding's decomposition of U-statistics (Hoeffding, 1948; Lee, 1990), can be found in the full paper [**AB-Conf3**]. The resulting bounds on the variance of $\widehat{U}_{h,n}$ are summarized in the following theorem.

Theorem 3.3 ([**AB-Conf3**]). *If $h(x, x') \in [0, 1]$ for all x, x', then*

$$\mathrm{Var}(\widehat{U}_{h,n}) \le \frac{1}{n(1-\beta)^2} + \frac{(1+\beta)^2}{2n(n-1)(1-\beta)^4}.$$

In order to achieve ϵ-LDP with a fixed k this becomes $\mathrm{Var}(\widehat{U}_{h,n}) \approx \frac{(1+k/\epsilon)^2}{n} + \frac{(1+k/\epsilon)^4}{2n^2} \approx \frac{k^2}{n\epsilon^2}$.

Continuous inputs. For U-statistics on discrete domains, the above strategy can be applied directly. More importantly however, it also leads to a natural protocol for the continuous case. As shown in Algorithm 3.3. the local randomizer first quantizes the input into k bins (for instance using simple or randomized rounding) before applying the previous procedure.

There are two sources of error in this protocol. The first one is due to the randomization needed to satisfy LDP in the quantized domain as bounded in Theorem 3.3. The second source of error is due to quantization. In order to control this error in a nontrivial way, we rely on an assumption on the kernel function h (namely, that it is Lipschitz) or the data distribution ω (namely, that it has Lipschitz density). Under these assumptions, we can bound the error with respect to the original domain by a term in $O(1/k^2)$. This leads to the following result.

Theorem 3.4 ([**AB-Conf3**]). *For simplicity, assume bounded domain $\mathcal{X} = [0, 1]$ and kernel values $h(x, y) \in [0, 1]$ for all $x, y \in \mathcal{X}$. Let Q correspond to simple rounding, $\epsilon > 0$, $k \ge 1$ and $\beta = k/(k + e^\epsilon - 1)$. Then Algorithm 3.3 satisfies ϵ-LDP. Furthermore:*

- *If h is L_h-Lipschitz in each of its arguments, then*

$$\mathbb{E}[(\widehat{U}_{h,n} - U_h)^2] \le \frac{1}{n(1-\beta)^2} + \frac{(1+\beta)^2}{2n(n-1)(1-\beta)^4} + \frac{L_h^2}{2k^2}.$$

- *If $d\omega/d\lambda$ is L_ω-Lipschitz w.r.t. some measure λ, then*

$$\mathbb{E}[(\widehat{U}_{h,n} - U_h)^2] \leq \frac{1}{n(1-\beta)^2} + \frac{(1+\beta)^2}{2n(n-1)(1-\beta)^4} + \frac{4L_\omega^2}{k^2} + \frac{4L_\omega^4}{k^4}.$$

Setting k so as to balance the quantization and estimation errors leads to the following corollary.

Corollary 3.1. *Under the conditions of Theorem 3.4, for $\epsilon \leq 1$ and large enough n, taking $k = n^{1/4}\sqrt{L\epsilon}$ leads to $\mathbb{E}[(\widehat{U}_{h,n} - U_h)^2] = O(L/\sqrt{n}\epsilon)$, where L corresponds to L_h or L_ω depending on the assumption.*

This result gives concrete error bounds for U-statistics whose kernel is Lipschitz, for arbitrary data distributions. One important example is the Gini mean difference, whose corresponding kernel $h(x_u, x_v) = |x_u - x_v|$ is 1-Lipschitz. On the other hand, for U-statistics with non-Lipschitz kernels, the data distribution must be sufficiently smooth (if not, it is easy to construct cases that make the algorithm fail).

Remark 3.5 (Combination with our decentralized algorithms)**.** *Algorithm 3.3 can be readily used to obtain locally differentially private versions of the decentralized estimation and optimization algorithms introduced in Section 3.1 and Section 3.2. This is achieved by running Algorithm 3.1 (resp. Algorithm 3.2) on the perturbed inputs $\tilde{x}_1, \ldots, \tilde{x}_n$ and computing unbiased estimates of $h(x_u, a_u)$ (resp. $\nabla h(\theta; x_u, a_u)$) using the formula in (3.10).*

3.3.3 Locally Private Protocol for Area under the ROC Curve

In this section, we describe an algorithm for computing the Area under the ROC curve (AUC). The AUC is a popular summary of the ROC curve which gives a single, threshold-independent measure of the classifier goodness: it corresponds to the probability that the predictor assigns a higher score to a randomly chosen positive point than to a randomly chosen negative one. AUCs are widely used as performance metrics in machine learning (Bradley, 1997) and have also been recently studied as fairness measures (Kallus and Zhou, 2019) [**AB-Conf2**]. Formally, let $\mathcal{X} \subset \mathbb{R}$. Each $x_u \in \mathcal{D}$ represent the score assigned to point u and is associated with a binary label $y_u \in \{-1, 1\}$. Let $\mathcal{D}^+ = \{s_u : y_u = 1\}$ and $\mathcal{D}^- = \{s_u : y_u = -1\}$ with $n^+ = |\mathcal{D}^+|$ and $n^- = |\mathcal{D}^-|$. The AUC is given by

$$\texttt{AUC} = \frac{1}{n^+ n^-} \sum_{s_u \in \mathcal{D}^+} \sum_{s_v \in \mathcal{D}^-} \mathbb{I}[s_u > s_v], \tag{3.11}$$

where $\mathbb{I}[\cdot]$ is the indicator function. Up to a $n(n-1)/2n^+n^-$ factor, it is easy to see that AUC is a U-statistic of degree 2 with kernel $h(x_u, x_v) = \mathbb{I}[s_u > s_v \wedge y_u > y_v] + \mathbb{I}[s_u < s_v \wedge y_u < y_v]$. Note that this kernel is discontinuous and therefore non-Lipschitz.

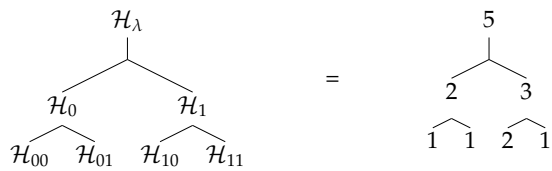


Figure 3.2: Hierarchical histogram $\mathcal{H}$ for multiset $\{0, 1, 2, 2, 3\}$ over the domain $\{0, 1, 2, 3\}$.

In the following, we assume $\mathcal{X}$ to be an ordered domain of size d, that is with each data point in $\{0, \ldots, d-1\}$. Note that all data is in practice discrete when represented in finite precision, so this is general. For simplicity of presentation we will assume that (i) $d = 2^\alpha$ for some integer α, and (ii) that the labels $y_1, \ldots, y_n$ are public.

Our solution for computing AUC in the local model relies on a hierarchical histogram construction that has been considered in previous works for private collection of high-dimensional data (Chan et al., 2012b), heavy hitters (Bassily et al., 2017), and range queries (Kulkarni et al., 2019). A hierarchical histogram is essentially a tree data structure on top of a histogram where each internal node is labeled with the sum of the values in the interval covered by it (see Figure 3.2). This allows to answer any range query about $\mathcal{X}$ by checking the value associated with $O(\log d)$ nodes in the tree.

Notation on trees. We represent a binary tree $\mathcal{H}$ of depth α with integer node labels as a total mapping from a prefix-closed set of binary strings of length at most α to the integers. We refer to the i-th node in level l of the tree by the binary representation of i padded to length l from the left with zeros. With this notation, $\mathcal{H}_\lambda$ is the label of the root node, as we use λ to denote the empty string, $\mathcal{H}_0$ (resp. $\mathcal{H}_1$) is the integer label of the left (resp. right) child of the root of $\mathcal{H}$, and in general $\mathcal{H}_p$ is the label of the node at path p from the root, i.e. the label of the node reached by following left or right children from the root according to the value of p (0 indicates left and 1 indicates right). Let b_i be the i-th node in the bottom level. For two binary strings $p, p' \in \{0,1\}^*$ we denote the prefix relation by $p' \preceq p$, and their concatenation as $p \cdot p'$.

Definition 3.2. *Let $\mathcal{D} = \{x_1, \ldots, x_n\}$ with $x_u \in \{0, \ldots, d-1\}$. A hierarchical histogram of $\mathcal{D}$ is a total mapping $\mathcal{H} : \{0,1\}^{\leq \log(d)} \to \mathbb{Z}$ defined as $\mathcal{H}(b) = |\{x \in \mathcal{D} \mid \exists b' \in \{0,1\}^* : b \cdot b' = b_x\}|$. For simplicity, we denote $\mathcal{H}(b)$ by $\mathcal{H}_b$.*

Algorithm. We use hierarchical histograms to compute AUC as follows. Let $\mathcal{H}^+$ and $\mathcal{H}^-$ be hierarchical histograms for $\mathcal{D}^+$ and $\mathcal{D}^-$. Note that $\mathcal{H}^+_\lambda = n^+$ and $\mathcal{H}^-_\lambda = n^-$. We can now define the unnormalized AUC, denoted UAUC, over hierarchical histograms recursively by letting $\mathtt{UAUC}(\mathcal{H}^+, \mathcal{H}^-, p)$ be 0 if p is a leaf, and otherwise setting:

$$\mathtt{UAUC}(\mathcal{H}^+, \mathcal{H}^-, p) = \mathcal{H}^+_{p \cdot 1} \mathcal{H}^-_{p \cdot 0} + \sum_{i \in \{0,1\}} \mathtt{UAUC}(\mathcal{H}^+, \mathcal{H}^-, p \cdot i) \ .$$

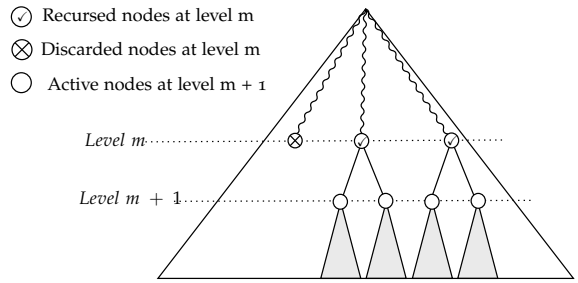


Figure 3.3: Our private AUC algorithm can be seen as a breath-first traversal of a tree, where at each level some nodes are selected for their subtrees to be explored further.

Thus we have $\texttt{AUC}(\mathcal{D}^+, \mathcal{D}^-) = \texttt{AUC}(\mathcal{H}^+, \mathcal{H}^-, \lambda) = \frac{1}{n^+ n^-}\texttt{UAUC}(\mathcal{H}^+, \mathcal{H}^-, \lambda)$. The above definition naturally leads to an algorithm that proceeds by traversing the trees $\mathcal{H}^+, \mathcal{H}^-$ top-down from the root λ, accumulating the products of counts from $\mathcal{H}^+, \mathcal{H}^-$ at nodes that correspond to entries in $\mathcal{H}^+$ that are bigger than entries in $\mathcal{H}^-$.

We now define a differentially private analogue. We assume the existence of an efficient frequency oracle which can be used to compute an LDP estimate $\hat{\mathcal{H}}$ of a hierarchical histogram $\mathcal{H}$ of n values in a domain of size 2^α with the following properties (i) $\hat{\mathcal{H}}$ is *unbiased*, (ii) $\mathrm{Var}(\hat{\mathcal{H}}) \leq V$, with V defined as $Cn\alpha$ for some small constant C (iii) the $\hat{\mathcal{H}}_p$ are pairwise independent and (iv) Each level of $\hat{\mathcal{H}}$ is independent of the other levels. Our private algorithm for computing an estimate of UAUC is then defined in terms of parameters n^+ and n^-, V^+ and V^- (bounding the variance of $\hat{\mathcal{H}}^+$ and $\hat{\mathcal{H}}^-$ respectively), and $a > 1$ is a small number depending on n^+, n^-, α and C.

Let $\tilde{\mathcal{H}}_p^\pm = \max(\hat{\mathcal{H}}_p^\pm, \sqrt{aV^\pm}/2)$, i.e. $\tilde{\mathcal{H}}_p^+ = \max(\hat{\mathcal{H}}_p^+, \sqrt{aV^+}/2)$ and $\tilde{\mathcal{H}}_p^- = \max(\hat{\mathcal{H}}_p^-, \sqrt{aV^-}/2)$. Given a threshold $\tau > 0$, our private estimate of UAUC is defined as follows. If p is a leaf then $\widehat{\texttt{UAUC}}(\hat{\mathcal{H}}^+, \hat{\mathcal{H}}^-, p)$ is 0, else if $\tilde{\mathcal{H}}_p^+ \tilde{\mathcal{H}}_p^- < \tau$ then it is given by $\frac{1}{2}\sum_{i\in\{0,1\}} \hat{\mathcal{H}}_{p\cdot i}^+ \sum_{i\in\{0,1\}} \hat{\mathcal{H}}_{p\cdot i}^-$. Otherwise, it is given recursively by

$$\hat{\mathcal{H}}_{p\cdot 1}^+ \hat{\mathcal{H}}_{p\cdot 0}^- + \sum_{i\in\{0,1\}} \widehat{\texttt{UAUC}}(\hat{\mathcal{H}}^+, \hat{\mathcal{H}}^-, p\cdot i).$$

As before, this definition leads to an algorithm. Note that the only difference with the non-private one is that it does not recurse into subtrees whose contribution to the UAUC is upper bounded sufficiently tightly. More concretely, the server starts by querying $\hat{\mathcal{H}}^+, \hat{\mathcal{H}}^-$ at the root, namely with $p = \lambda$. If p is a leaf then we return 0 as the AUC. Otherwise, the algorithm checks whether $\tilde{\mathcal{H}}_p^+ \tilde{\mathcal{H}}_p^- < \tau$. If so, then the algorithm concludes that there is not much to gain in exploring the subtrees rooted at $p \cdot 0$ and $p \cdot 1$, and returns $\frac{1}{2}\sum_{i\in 0,1} \hat{\mathcal{H}}_{p\cdot i}^+ \sum_{i\in 0,1} \hat{\mathcal{H}}_{p\cdot i}^-$ as an estimate of $\frac{1}{2}\mathcal{H}_p^+ \mathcal{H}_p^-$. In this case we call p a *discarded* node. On the other hand, if $\tilde{\mathcal{H}}_p^+ \tilde{\mathcal{H}}_p^- \geq \tau$, the algorithm proceeds as its non-private analogue, accumulating the contribution to the UAUC from the direct subtrees of p and recursing into nodes $p \cdot 0$ and $p \cdot 1$. In this case we refer to p as a *recursed* node. Thus every node

$p \in \{0,1\}^{\leq \alpha}$ will be either recursed, a leaf or there will be a discarded node p' such that $p' \preceq p$. This is depicted in Figure 3.3.

Utility guarantee. Our algorithm has two sources of error: (i) the error in privately estimating the contribution of the recursed nodes to UAUC, and (ii) the one incurred by discarding nodes. The following result bounds the error of our protocol, where the threshold τ has been carefully chosen to balance these two errors.

Theorem 3.5 ([AB-Conf3]). *Consider $\alpha \leq \sqrt{n}$ and assume that the following holds:*

1. $\mathbb{E}[\hat{\mathcal{H}}_p^{\pm} - \mathcal{H}_p^{\pm}] = 0$, *i.e., frequency estimates are unbiased.*
2. $\mathbb{E}[(\hat{\mathcal{H}}_p^{\pm} - \mathcal{H}_p^{\pm})^2] \leq V^{\pm} = Cn^{\pm}\alpha$, *i.e., frequency estimates have bounded squared error.*
3. *For distinct $p, p' \in \{0,1\}^{\leq \alpha}$ with $|p| = |p'|$, $\hat{\mathcal{H}}_p^{\pm}$ and $\hat{\mathcal{H}}_{p'}^{\pm}$ are independent, i.e., the frequency estimates are pairwise independent.*
4. $\forall m \leq \log(d)$, *the lists $(\hat{\mathcal{H}}_p^{\pm})_{p \in \{0,1\}^{\leq m}}$ and $(\hat{\mathcal{H}}_p^{\pm})_{p \in \{0,1\}^{>m}}$ are independent of each other.*

Then, we have:

$$\mathbb{E}[(\widehat{\mathtt{UAUC}} - \mathtt{UAUC})^2] \leq Cn^- n^+ \alpha^2 \Big(2n + (4a+1)\min(n^-, n^+) + \frac{21\sqrt{2nC\alpha}}{\sqrt{a}-1}\Big).$$

Sketch of proof. Our proof crucially relies on conditioning on previous levels when bounding the error at a given level. To illustrate this, we briefly outline here how we bound the error due to recursed nodes. For any $m \in [\![\alpha]\!]$, let R^m be the set of nodes recursed on at level m and $E_m^R = \sum_{p \in R^{m-1}} \hat{\mathcal{H}}_{p\cdot 1}^{+} \hat{\mathcal{H}}_{p\cdot 0}^{-} - \mathcal{H}_{p\cdot 1}^{+} \mathcal{H}_{p\cdot 0}^{-}$ be the contribution to the error by recursed nodes at level m. We would like to bound the expected squared error $\mathbb{E}[{E^R}^2] = \mathbb{E}[\sum_{m \in [\![\alpha]\!]} \sum_{m' \in [\![\alpha]\!]} E_m^R E_{m'}^R]$ incurred by recursed nodes across all levels. While the frequency oracle is unbiased, E_m^R and $E_{m'}^R$ are not independent. Nevertheless, we can show that for $m' > m$, $\mathbb{E}[E_m^R E_{m'}^R] = 0$ because the conditional expectation of $E_{m'}^R$ with respect to the answers of the frequency oracle up to level m' is 0, i.e., $E_1^R, \ldots, E_{m'}^R$ is a martingale difference sequence. Thus, we have $\mathbb{E}[{E^R}^2] = \sum_{m \in [\![\alpha]\!]} \mathbb{E}[{E_m^R}^2]$. We then bound the expected value of $|R^m|$ by a quantity B that is independent of m, allowing us to bound $\mathbb{E}[{E_m^R}^2]$, for any m, in terms of B. □

Instantiating $\hat{\mathcal{H}}$. Theorem 3.5 does not yield a complete algorithm as it does not specify an frequency oracle for computing estimates $\hat{\mathcal{H}}$ of a hierarchical histogram with the required properties. Such a frequency oracle can be instantiated using the protocol proposed by Kulkarni et al. (2019) in the context of range queries. In [AB-Conf3], we show how to improve the communication and space complexity of this solution by combining this protocol with ideas from Bassily et al. (2017), in particular the use of the Hadamard transform. This leads to the following result.

Theorem 3.6. *There is a one-round non-interactive protocol in the local model which achieves* $\mathbb{E}[(\widehat{\mathtt{UAUC}} - \mathtt{UAUC})^2] = O(\alpha^2 \log(1/\delta)/n\epsilon^2)$ *under* (ϵ, δ)*-LDP and* $\mathbb{E}[(\widehat{\mathtt{UAUC}} - \mathtt{UAUC})^2] = O(\alpha^3/n\epsilon^2)$ *under* ϵ*-LDP. Every user submits one bit, and the server does* $O(n \log d)$ *computation and requires* $O(\log d)$ *additional reconstruction space.*

3.3.4 Beyond Local DP: Protocols from 2-Party Secure Computation

So far, we have proposed a specialized LDP protocol for the AUC, and a generic LDP protocol which requires some assumption on the kernel function or the data distribution to guarantee nontrivial error bounds. We conjecture that no LDP protocol can guarantee nontrivial error for arbitrary kernels and distributions (we leave this as an open question).

In this section, we consider a specific relaxation of LDP by allowing pairs of users u and v to compute a randomized version $\tilde{h}(x_u, x_v)$ of their kernel value $h(x_u, x_v)$ with 2-party secure computation (2PC).[4] Unsurprisingly, by using advanced composition theorems (see Appendix A), we can easily show that in this relaxed model we can match the MSE of $O(\frac{\ln(1/\delta)}{n\epsilon^2})$ achievable for computing regular (univariate) averages in the (ϵ, δ)-LDP model. However, such a protocol requires $O(n^2)$ communication as all pairs of users need to compute $\tilde{h}(x_u, x_v)$ via 2PC, and does not satisfy pure ϵ-DP.

Proposed protocol. To address these limitations, we propose that the aggregator asks only a (random) subset of pairs of users (u, v) to submit their randomized kernel value $\tilde{h}(x_u, x_v)$. The idea is to trade-off between the error due to privacy (which increases as more pairs are used due to composition) and the *subsampling error* (for not averaging over all pairs). Given a positive integer P (which should be set to a small constant independent of n) and assuming n to be even for simplicity, we propose the following protocol:

1. *Subsampling:* The aggregator samples P independent permutations $\mathfrak{s}_1, \ldots, \mathfrak{s}_P \in \mathfrak{S}_n$ of the set of users $\{1, \ldots, n\}$. This defines a (multi)set of $Pn/2$ pairs $\mathcal{P} = \{(\mathfrak{s}_p(2u-1), \mathfrak{s}_p(2u))\}_{p \in [P], 1 \leq u \leq n/2}$.

2. *Perturbation:* For each pair of users $(u, v) \in \mathcal{P}$, users compute $\tilde{h}(x_u, x_v)$ via 2PC and sends it to the aggregator.

3. *Aggregation:* The aggregator computes an estimate of $\widehat{U}_{h,n}$ as a function of $\{\tilde{h}(x_u, x_v)\}_{(u,v) \in \mathcal{P}}$.

Analysis. We have the following result for the Laplace mechanism applied to real-valued kernel functions (the extension to randomized response for discrete-valued kernels is straightforward). The proof relies on an exact characterization of the subsampling error by leveraging results on the variance of incomplete U-statistics (Blom, 1976).

[4] In practice, this implies that privacy guarantees only hold against computationally bounded adversaries.

Theorem 3.7 ([**AB-Conf3**]). *Let $\epsilon > 0$, $P \geq 1$ and assume that the kernel h has values in $[0,1]$. Consider our subsampling protocol above with $\tilde{h}(x_u, x_v) = h(x_u, x_v) + \eta_{u,v}$ where $\eta_{u,v} \sim Lap(P/\epsilon)$, and $\widehat{U}_{h,n} = \frac{2}{Pn}\sum_{(u,v)\in\mathcal{P}} \tilde{h}(x_u, x_v)$. Then the protocol satisfies ϵ-DP with a total communication cost of $O(Pn)$. Moreover:*

$$\mathbb{E}[(\widehat{U}_{h,n} - U_h)^2] = \frac{2}{Pn}\Big(2(P-1)(1-\frac{1}{n-1})\zeta_1 + (1+\frac{P-1}{n-1})\zeta_2\Big) + \frac{2P}{n\epsilon^2},$$

where $\zeta_1 = \mathrm{Var}(h(x_1, X_2) \mid x_1))$ and $\zeta_2 = \mathrm{Var}(h(X_1, X_2))$.

The error in Theorem 3.7 is of $O(\frac{1}{Pn} + \frac{P}{n\epsilon^2})$. Remarkably, this shows that the $O(1/n)$ variance of the estimate that uses all pairs is preserved when subsampling only $O(n)$ pairs. This is made possible by the strong dependence structure in the $O(n^2)$ terms of the original U-statistic. As expected, P rules a trade-off between the errors due to subsampling and to privacy: the larger P, the smaller the former but the larger the latter (as each user must split its budget across P pairs). The optimal value of P depends on the kernel function and the data distribution (through ζ_1 and ζ_2) on the one hand, and the privacy budget ϵ on the other hand. In practice, P can be set to a small constant.

Implementing 2PC. Securely computing the randomized kernel value $\tilde{h}(x_u, x_v)$ can be done efficiently for many kernel functions and local randomizers of interest, as the number of parties involved is limited to 2 and parties are honest-but-curious. A suitable 2PC technique are garbled circuits (Yao, 1986; Lindell and Pinkas, 2009; Evans et al., 2018), which are well-suited to compute Boolean comparisons involved in the kernels of many U-statistics of interest (e.g., AUC). The circuits for computing the kernels can then be extended with output perturbation following ideas from Dwork et al. (2006a) and Champion et al. (2019). We refer to [**AB-Conf3**] for details on design and complexity.

3.4 Experiments

In this section, we briefly illustrate the practical behavior of the proposed algorithms. More results and details can be found in [**AB-Conf21**]; [**AB-Conf16**]; [**AB-Conf3**].

Decentralized pairwise estimation. We focus on the decentralized estimation of the AUC (3.11) on the SVMguide3 binary classification dataset which contains $n = 1260$ points in 23 dimensions.[5] The scores are given by a simple linear scoring rule $s_\theta(x) = \theta^\top x$ with θ set to the difference between the class means. We perform our simulations on three types of network graphs of increasing connectivity: a 2D grid, a Watts-Strogatz random graph (Watts and Strogatz, 1998) and the complete graph. For each network, we perform 50 runs of GoSta (Algorithm 3.1) and U2-gossip, a simple baseline which does not use

[5] http://mldata.org/repository/data/viewslug/svmguide3/

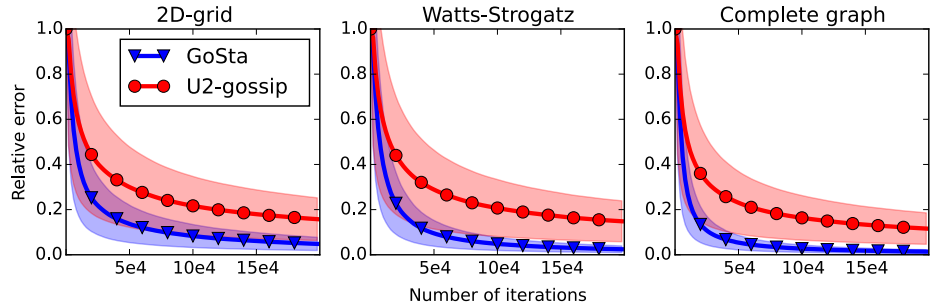
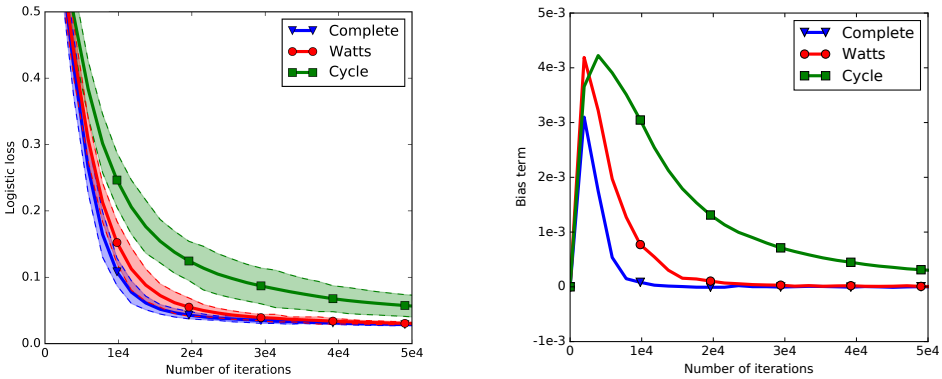


Figure 3.4: Decentralized AUC estimation on the SVMguide3 dataset.



(a) Evolution of the objective function.

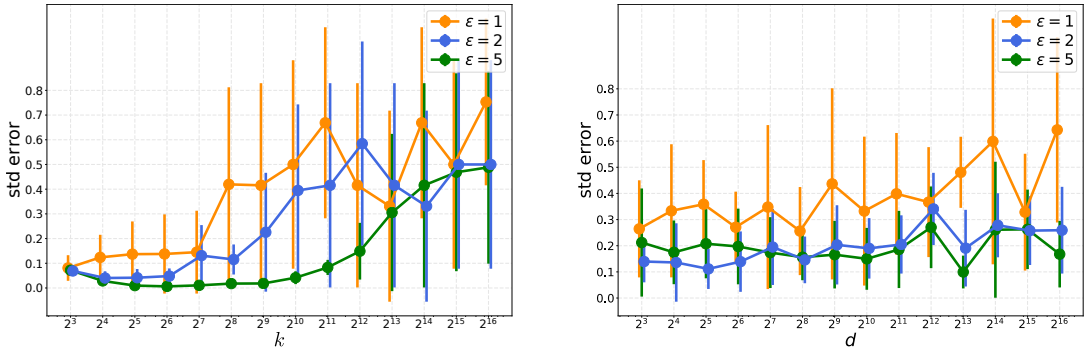
(b) Evolution of the bias term.

Figure 3.5: Decentralized AUC maximization on Breast cancer dataset. Best seen in color.

averaging and instead relies on a double data propagation step so that each user may observe all pairs of data points (Pelckmans and Suykens, 2009). Figure 3.4 shows the evolution over time of the average relative error (solid lines) and the associated standard deviation *across users* (filled areas) for both algorithms on each type of network. As suggested by our theoretical analysis, GoSta converges faster on more connected networks. Furthermore, it outperforms U2-gossip in all cases and the variance of its estimates across users is also lower, thanks to the averaging step.

Decentralized pairwise optimization. We focus on learning a linear scoring rule $s_\theta(x) = \theta^\top x$ by AUC maximization on the Breast Cancer dataset, which consists of $n = 699$ points in 11 dimensions.[6] We use the loss function defined in (3.5) and do not apply any regularization (i.e., $\psi \equiv 0$). We perform our simulations on the same types of graphs as for decentralized estimation, except that we use a cycle instead of a 2D grid. We run the asynchronous version of Algorithm 3.2 (see Remark 3.4) with step size $\gamma(t) = 1/\sqrt{t}$ over 50 runs. Figure 3.5a shows the evolution of the objective (solid

[6] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)



(a) Generic LDP protocol.

(b) AUC-specific protocol.

Figure 3.6: Private estimation of the AUC of a logistic regression model trained on the Diabetes dataset. Mean and standard deviations of the errors are over 20 runs.

lines) and the associated standard deviation across users (filled areas) across iterations. Again and as expected, the convergence rate on the complete and the Watts-Strogatz networks is much better than on the cycle network. The standard deviation across users also decreases with the connectivity of the network. The good empirical convergence of our algorithm comes from the fact that the bias term $\bar{\epsilon}^n(t)^\top \pi_t(\bar{z}^n(t)$ in Theorem 3.2 vanishes quite fast, as shown in Figure 3.5b. Moreover, its order of magnitude is negligible compared to the objective.

Private pairwise estimation. We first present results on private estimation of the AUC. We use the Diabetes dataset for the binary classification task of determining whether a patient will be readmitted in the next 30 days after being discharged.[7] We train a logistic regression model $s : \mathcal{X} \to [0, 1]$ which is used to score data points, and apply our protocol to privately compute the AUC on the test set. Patients readmitted before 30 days form the positive class, which is also the minority class ($n^+ = 693$ and $n^- = 95985$). We do not consider the class information to be sensitive, as opposed to the score $s(x)$ computed on private user data x which includes detailed medical information. Figure 3.6a shows the results of our generic LDP protocol (Algorithm 3.3). On this dataset, a fully trained logistic regression model yields scores of positive and negative points that are well separated. Hence, even the kernel is not Lipschitz, data can be quantized to a sufficiently small k and the protocol is able to achieve small error. However, in practice this is difficult to assess in advance, and we see that the error blows up for larger k. In contrast, Figure 3.6b shows that our AUC-specific protocol introduced in Section 3.3.3 can work well even when the domain size d (obtained by discretization of $[0, 1]$ into $\{0, \ldots, d-1\}$) is large. Although the generic LDP protocol can achieve lower error on this dataset, the better scalability of our AUC protocol with the number of bins makes it very useful for problems that require

[7] https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008

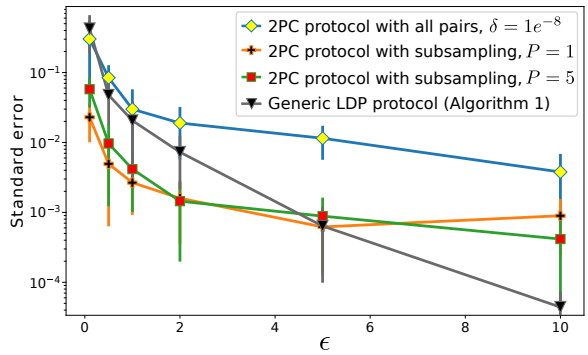


Figure 3.7: Private estimation of Kendall's tau coefficient on Tripadvisor dataset. Mean and standard deviation of the errors are over 20 runs.

fine-grained discretization (see [**AB-Conf3**] for concrete examples).

To illustrate the utility gains that can be obtained with our 2PC protocol introduced in Section 3.3.4, we present results on private estimation of Kendall's tau coefficient, which measures the ordinal association between two random variables X and X'. Let $\mathcal{X} \subset \mathbb{R}^2$ and $\mathcal{D} = \{((x_1, x'_1), \ldots, (x_n, x'_n))\}$. Kendall's tau coefficient can be written as a U-statistic (3.7) with kernel $h((x_u, x'_u), (x_v, x'_v)) = \text{sign}(x_u - x_v)\,\text{sign}(x'_u - x'_v)$. We use the Tripadvisor dataset, which consists of discrete user ratings (from scale -1 to 5) for hotels in San Francisco over many service quality metrics such as room service, location, room cleanliness, front desk service etc.[8] After discarding the records with missing values, we have over 246K records. Let (x_u, x'_u) be ratings given by user u to the room (x_u) and the cleanliness (x'_u). We compare the privacy-utility trade-off of our generic LDP protocol (Algorithm 3.3 without quantization, since inputs can take only 36 values), our 2PC protocol based on subsampling, and the naive 2PC protocol that computes all pairs and relies on advanced composition. The results shown in Figure 3.7 show that the naive 2PC protocol performs worst due to composition. Our generic LDP protocol performs better thanks to the small domain size. Finally, our 2PC protocol with subsampling achieves the lowest error by roughly an order of magnitude in high privacy regimes ($\epsilon \leq 2$) while keeping the communication cost linear in n. As predicted by our analysis, $P = 1$ is best in high privacy regimes, where the error due to privacy dominates the subsampling error. We also see that $P > 1$ can be used to reduce the overall error in low privacy regimes. This result show that if one is willing to slightly relax the LDP model to allow pairwise communication among users and the kernel can be computed efficiently via 2PC, our third protocol performs best in terms of accuracy.

[8] http://www.preflib.org/data/combinatorial/trip/

Chapter 4

Better Privacy-Utility Trade-offs for Decentralized Learning

The adoption of decentralized learning crucially depends on the ability to deliver accurate models (*utility*) while protecting the *privacy* of users against untrusted participants involved in the learning process. To guarantee privacy for the approaches presented in the previous chapters, we have relied on (variants of) the notion of local differential privacy (LDP) (Kasiviswanathan et al., 2008; Duchi et al., 2013). LDP protects users against an adversary that can observe everything except their local memory, hence it requires that each user randomizes his/her contributions locally before sharing them. This very strong model of privacy comes at a significant cost in utility: for real summation with n users, the best possible error under LDP is a factor $\sqrt{n}$ larger than in the centralized model of DP, where a trusted curator handles the raw data (Chan et al., 2012a; Chen et al., 2020). Large utility gaps between LDP and the trusted curator model are also known to hold for machine learning (see e.g., Zheng et al., 2017; Wang et al., 2018).

In this chapter, we study relaxations of the local model of differential privacy in which users can communicate through *secure channels*. Each user thus seeks to protect from an adversary (e.g., a coalition of other users or an untrusted aggregator) who can only observe a subset of the messages. We propose two novel approaches that leverage secure channels to *provably improve the privacy-utility trade-off of decentralized learning* compared to local DP, and in some cases even match the utility of the trusted curator model.

Our first contribution is targeted at decentralized learning with an untrusted aggregator, where the key communication step is to privately aggregate model updates from all (or a large number of) users. Indeed, popular federated learning algorithms (see e.g., McMahan et al., 2017; Li et al., 2020; Karimireddy et al., 2020; Fallah et al., 2020; Hanzely et al., 2020) that minimize a sum of local objectives of the form (1.2) all follow the same high-level procedure: at round t, each user u computes a local update model $\theta_u(t)$ based on $\mathcal{D}_u$ and the current global model $\theta(t-1)$, and an untrusted aggregator computes a new global model as $\theta(t) = \frac{1}{n}\sum_u \theta_u(t)$. We thus focus on the problem of decentralized

differentially private averaging under potentially colluding and malicious users. For this challenging setting, we propose a protocol, called Gopa, in which pairs of users securely exchange some correlated Gaussian noise terms along the edges of a network graph so as to mask their private values without affecting the global average. Remarkably, we establish that our approach can *achieve nearly the same privacy-utility trade-off as a trusted curator* who would average the values of honest (non-colluding) users. Furthermore, each user needs only to *communicate with a logarithmic number of other users*, making the protocol scalable to large numbers of users. Finally, leveraging standard cryptographic primitives like commitment schemes and zero knowledge proofs, we can make Gopa verifiable by providing users with the means of proving the correctness of their computations without compromising the scalability or the privacy guarantees of the protocol.

Our second contribution is targeted at fully decentralized learning algorithms and aims to show that *the lack of central coordinator can amplify privacy guarantees*. To this end, we introduce *network differential privacy*, a novel relaxation of LDP that naturally arise in the fully decentralized setting and effectively captures the fact that each user only observes information received from her/his neighbors. Under this relaxation, we study a decentralized model of computation where a token containing the current estimate performs a walk on the network graph and is updated sequentially by the user who receives it. We start by analyzing the case of a walk over a directed ring and propose simple algorithms for computing real summations and discrete histograms which achieve a privacy gain of $O(1/\sqrt{n})$ compared to LDP, thereby *matching the privacy-utility trade-off of a trusted aggregator*. We then consider the case of random walks over a complete graph. We provide an algorithm for real summation and prove a privacy amplification result of $O(1/n^{1/3})$ compared to the same algorithm analyzed under LDP. We then propose a decentralized SGD algorithm that achieves a privacy amplification of $O(\ln n/\sqrt{n})$, nearly matching the utility of *centralized* private SGD (Song et al., 2013; Bassily et al., 2014). Interestingly, our two algorithms can tolerate a constant number of collusions at the cost of a reduction in the privacy amplification effect. To the best of our knowledge, our work is the first to show that formal privacy gains can be naturally obtained from full decentralization (i.e., from having no central coordinator). Our results imply that the privacy guarantees of some fully decentralized algorithms have been largely underestimated, providing a new incentive for using such approaches beyond the usual motivation of scalability.

The work covered in this chapter is available in two recent preprints [**AB-Preprint7**]; [**AB-Preprint6**].

Related work. Our contributions belong to the recent line of work which attempts to relax the local DP model so as to improve utility without relying on a trusted curator. This is usually achieved through the use of cryptographic primitives.[1] A popular ap-

[1]Strictly speaking, relying on such primitives (including the use of secure channels) typically introduces an additional assumption of computationally bounded adversaries. The resulting privacy notion is some-

proach is to rely on *secure aggregation* to average the individual contributions of all users, see (Dwork et al., 2006a; Shi et al., 2011; Ács and Castelluccia, 2011; Bonawitz et al., 2017; Chan et al., 2012b) for protocols and (Jayaraman et al., 2018) for a concrete example of application to machine learning. While secure aggregation allows in principle to recover the utility of the trusted curator model, it suffers from two main drawbacks. First, existing protocols require $\Omega(n^2)$ total communication, which is hardly feasible beyond a few hundred users. In contrast, we propose a protocol which requires only $O(n \log n)$ communication.[2] Second, combining secure aggregation with DP is nontrivial as the noise must be added in a distributed fashion. Existing complete systems (Kairouz et al., 2021a) assume an ideal secure aggregation functionality which does not reflect the impact of colluding/malicious users. In these more challenging settings, it is not clear how to add the necessary noise for DP and what the resulting privacy/utility trade-offs would be.

Recent work has also considered passing user contributions through a *secure shuffler* to obfuscate the source of the messages, giving rise to the so-called shuffle model of DP (Cheu et al., 2019; Erlingsson et al., 2019; Balle et al., 2019; Balle et al., 2020; Ghazi et al., 2020; Feldman et al., 2020). The shuffle model allows to match the trusted curator utility for some tasks, including differentially private averaging (Balle et al., 2020). However, practical implementations of secure shuffling are not discussed in the above work. Existing solutions typically rely on multiple layers of routing servers (Dingledine et al., 2004) with high communication overhead and non-collusion assumptions.

Local DP is convenient to work with in fully decentralized algorithms as random perturbations are applied locally by each user. Despite its cost in utility, local DP has thus been the standard model in existing work on private fully decentralized (see e.g. Huang et al., 2015; Li et al., 2018; Cheng et al., 2019; Zhang et al., 2018; Xu et al., 2020). This is also the approach we have taken in Chapter 2 and Chapter 3. Note that matching the trusted curator utility with the secure aggregation or shuffling primitives discussed above would require all users to interact with each other at each step and/or to rely on a central coordinator. These solutions thus appear to be incompatible with full decentralization.

A related line of work has studied mechanisms that "amplify" the DP guarantees of a private algorithm. Beyond privacy *amplification by shuffling* based on the shuffling primitive mentioned above (Erlingsson et al., 2019; Balle et al., 2019; Feldman et al., 2020), we can cite *amplification by subsampling* (Balle et al., 2018) and *amplification by iteration* (Feldman et al., 2018). These schemes are generally difficult to apply in a federated/decentralized setting: the former requires that the identity of subsampled participants remain secret, while the latter assumes that only the final model is revealed. Our work actually allows to leverage these results in a novel context: this is made possible by the restricted view of participants offered by fully decentralized algorithms and captured

times referred to as *computational DP* (Mironov et al., 2009).

[2]We note that, independently and in parallel to our work, Bell et al. (2020b) recently proposed a secure aggregation protocol with $O(n \log n)$ communication.

by our notion of network DP.

A original aspect of our contributions is to match the privacy-utility trade-off of the trusted curator model without resorting to the functionalities of secure aggregation or secure shuffling. We are not aware of other work sharing this feature.

Outline of the chapter. This chapter consists of two parts: Section 4.1 presents our protocol for differentially private averaging, while Section 4.2 studies how fully decentralized protocols can amplify privacy guarantees.

4.1 An Accurate, Scalable and Verifiable Protocol for Decentralized Differentially Private Averaging

In this section, we focus on the problem of decentralized differentially private averaging, an essential building block of federated learning algorithms. Section 4.1.1 describes the problem setting. In Section 4.1.2, we present a scalable protocol in which users exchange correlated Gaussian noise along the edges of a network graph, complemented by independent noise added by each user. In Section 4.1.3, we analyze the DP guarantees of our protocol, showing that we can nearly match the utility of the trusted curator model with only logarithmic communication per user. Finally, Section 4.1.4 presents a method to ensure the correctness of our protocol against malicious users without compromising the efficiency and privacy guarantees of the protocol.

4.1.1 Problem Setting

We denote the set of users by $U = [\![n]\!]$. Each user $u \in U$ holds a private value x_u, which can be thought of as being computed from his/her private dataset $\mathcal{D}_u$. For simplicity, we assume that x_u lies in a bounded interval of $\mathbb{R}$ (without loss of generality, we assume $x_u \in [0,1]$). The extension to the vector case is straightforward. We denote by $X = [x_1, \ldots, x_n]^\top \in [0,1]^n$ the column vector of private values.

Users communicate over a network represented by a connected undirected graph $\mathcal{G} = (U, E)$, where $\{u, v\} \in E$ indicates that users u and v are neighbors in $\mathcal{G}$ and can exchange secure messages. For a given user u, we denote by $\mathcal{N}(u) = \{v : \{u, v\} \in E\}$ the set of its neighbors. We note that in settings where users can only communicate through a central server, the latter can act as a relay that forwards (encrypted and authenticated) messages between users, as done in secure aggregation (Bonawitz et al., 2017).

The users aim to collaboratively estimate the average $x^{avg} = \frac{1}{n}\sum_{u=1}^{n} x_u$ without revealing their individual private values. Such a protocol can be readily used to privately execute decentralized learning algorithms that interact with data through global averages over values computed locally by the participants, but do not actually need to see the

individual values. As discussed in the introduction of this chapter, most decentralized learning algorithms which rely on a central coordinator fall into this category.

Threat model. In this part, we consider two commonly adopted adversary models formalized by Goldreich (1998). A *honest-but-curious* (*honest* for short) user will follow the protocol specification, but may use all the information obtained during the execution to infer information about other users. A honest user may accidentally drop out at any point of the execution (in a way that is independent of the private values X). On the other hand, a *malicious user* may deviate from the protocol execution (e.g, sending incorrect values or dropping out on purpose). Malicious users can collude, and thus will be seen as a single malicious party (the *adversary*).

We want our protocol to satisfy differential privacy, where the output of our protocol in the sense of Definition 1.1 will correspond to the view of the adversary (we will define this explicitly in Section 4.1.3). Our DP guarantees will hold under the assumption that honest users communicate through secure channels. In Section 4.1.4, we will also provide *correctness guarantees* for our protocol that will hold under some form of the Discrete Logarithm Assumption (DLA), a standard assumption in cryptography (see Chapter 7 of Katz and Lindell, 2014).

For a given execution of the protocol, we denote by U^O the set of the users who remained online until the end (i.e., did not drop out). Users in U^O are either honest or malicious: we denote by $U^H \subseteq U^O$ those who are honest, by $n_H = |U^H|$ their number and by $\rho = n_H/n$ their proportion with respect to the total number of users. We also denote by $\mathcal{G}^H = (U^H, E^H)$ the subgraph of $\mathcal{G}$ induced by the set of honest users U^H, i.e., $E^H = \{\{u, v\} \in E : u, v \in U^H\}$. The properties of $\mathcal{G}$ and $\mathcal{G}^H$ will play a key role in the privacy and scalability guarantees of our protocol.

4.1.2 Proposed Protocol

In this section we describe our protocol called Gopa (GOssip noise for Private Averaging). The high-level idea of Gopa is to have each user u mask its private value by adding two different types of noise. The first is a sum of pairwise-correlated noise terms $\Delta_{u,v}$ over the set of neighbors $v \in \mathcal{N}(u)$ such that each $\Delta_{u,v}$ cancels out with the $\Delta_{v,u}$ of user v in the final result. The second type of noise is an independent term η_u which does not cancel out. At the end of the protocol, each user has generated a noisy version $\hat{x}_u$ of its private value x_u, which takes the form:

$$\hat{x}_u = x_u + \sum_{v \in N(u)} \Delta_{u,v} + \eta_u. \tag{4.1}$$

Algorithm 4.1 presents the detailed steps. Neighboring nodes $\{u, v\} \in E$ contact each other to draw a real number from the Gaussian distribution $\mathcal{N}(0, \sigma_\Delta^2)$, that u adds to its

Algorithm 4.1 GOPA protocol for decentralized differentially private averaging.

Public Parameters: Network graph $\mathcal{G} = (U, E)$, noise scales $\sigma_\Delta^2, \sigma_\eta^2 \in \mathbb{R}^+$
Input: Private value $x_u \in [0, 1]$ for each user $u \in U$
1: **for all** neighbor pairs $\{u, v\} \in E$ s.t. $u < v$ **do**
2: Users u and v draw a random $\eta_{u,v} \sim \mathcal{N}(0, \sigma_\Delta^2)$ and set $\Delta_{u,v} \leftarrow \eta_{u,v}$, $\Delta_{v,u} \leftarrow -\eta_{u,v}$
3: **for all** users $u \in U$ **do**
4: User u draws $\eta_u \sim \mathcal{N}(0, \sigma_\eta^2)$ and reveals noisy value $\hat{x}_u \leftarrow x_u + \sum_{v \in N(u)} \Delta_{u,v} + \eta_u$

private value and v subtracts. Intuitively, each user thereby distributes noise masking its private value across its neighbors so that even if some of them are malicious and collude, the remaining noise values will be enough to provide the desired privacy guarantees. The idea is reminiscent of uniformly random pairwise masks in secure aggregation (Bonawitz et al., 2017) but we use Gaussian noise and restrict exchanges to the edges of the graph instead of generating masks for all pairs of users. As in gossip protocols (see our algorithms of Chapter 3), the pairwise exchanges can be performed asynchronously and in parallel. Additionally, every user $u \in U$ adds an independent noise term $\eta_u \sim \mathcal{N}(0, \sigma_\eta^2)$ to its private value. This noise will ensure that the final estimate of the average satisfies differential privacy (see Section 4.1.3).

Utility of GOPA. The protocol generates a set of noisy values $\hat{X} = [\hat{x}_1, \ldots, \hat{x}_n]^\top$ which can then be publicly released to an untrusted aggregator. The estimated average is given by $\hat{X}^{avg} = \frac{1}{n} \sum_{u \in U} \hat{x}_u = x^{avg} + \frac{1}{n} \sum_{u \in U} \eta_u$, which has expected value x^{avg} and variance σ_η^2 / n. Recall that the local model of DP, where each user releases a locally perturbed input without communicating with other users, would require $\sigma_\eta^2 = O(1)$. In contrast, we would like the total amount of independent noise to be of order $O(1/n_H)$ as needed to protect the average of honest users with the Gaussian mechanism in the trusted curator model (Dwork and Roth, 2014). We will show in Section 4.1.3 that we can achieve this by choosing an appropriate graph and pairwise variance σ_Δ^2.

Dealing with dropout. A user $u \notin U^O$ who drops out during the execution of the protocol does not actually publish any noisy value (i.e., $\hat{x}_u$ is empty). The estimated average is thus computed by averaging only over the noisy values of users in U^O. Additionally, any residual noise term that a user $u \notin U^O$ may have exchanged with a user $v \in U^O$ before dropping out can be "rolled back" by having v reveal $\Delta_{u,v}$ so it can be subtracted from the result (we will ensure this does not threaten privacy by having sufficiently many neighbors, see Section 4.1.3). We can thus obtain an estimate of $\frac{1}{|U^O|} \sum_{u \in U^O} x_u$ with variance $\sigma_\eta^2 / |U^O|$. Note that even if some residual noise terms are not rolled back, e.g. to avoid extra communication, the estimate remains unbiased (with a larger variance that depends on σ_Δ^2). This is a rather unique feature of GOPA which comes from the use of Gaussian

noise rather than the uniformly random noise used in secure aggregation (Bonawitz et al., 2017). We refer to [**AB-Preprint7**] for details on strategies to handle drop out.

4.1.3 Differential Privacy Guarantees

In this section, we prove differential privacy guarantees for Gopa. We start by defining the view of the adversary, i.e., the knowledge acquired by colluding malicious users during a given execution of the protocol. It consists of the following: (i) the noisy value $\hat{x}_u$ of all users $u \in U^O$ who did not drop out, (ii) the private value x_u and the noise η_u of the malicious users, and (iii) all $\Delta_{u,v}$'s for which u or v is malicious. We also assume that the adversary knows the full network graph $\mathcal{G}$ and all the pairwise noise terms exchanged by dropped out users (since they may be rolled back, as explained in Section 4.1.2). The only unknowns are thus the private value x_u and independent noise η_u of each honest user $u \in U^H$, as well as the $\Delta_{u,v}$'s exchanged between honest users $\{u, v\} \in E^H$. Letting $N^H(u) = \{v : \{u, v\} \in E^H\}$, from the above knowledge the adversary can subtract $\sum_{v \in N(u) \setminus N^H(u)} \Delta_{u,v}$ from $\hat{x}_u$ to obtain $\hat{x}_u^H = x_u + \sum_{u \in N^H(u)} \Delta_{u,v} + \eta_u$ for every honest $u \in U^H$. The view of the adversary can thus be summarized by the vector $\hat{X}^H = (\hat{x}_u^H)_{u \in U^H}$ and the correlation between its elements.

Now, adapting differential privacy (Definition 1.3) to our setting, for any input X and any possible outcome $\hat{X}$, we need to compare the probability of the outcome being equal to $\hat{X}$ when a (non-malicious) user $v_1 \in U$ participates in the computation with private value $x_{v_1}^A$ to the probability of obtaining the same outcome when the value of v_1 is exchanged with an arbitrary value $x_{v_1}^B \in [0, 1]$. Since honest users drop out independently of X, in our analysis we will fix an execution of the protocol where some set U^H of n_H honest users have remained online until the end of the protocol. For notational simplicity, we denote by X^A the vector of private values $(x_u)_{u \in U^H}$ of these honest users in which a user v_1 has value $x_{v_1}^A$, and by X^B the vector where v_1 has value $x_{v_1}^B$. X^A and X^B differ in only in the v_1-th coordinate, and their maximum difference is 1.

Privacy guarantees In [**AB-Preprint7**], we show that Gopa can match the privacy-utility trade-off of the trusted curator setting as long as $\mathcal{G}^H$ (the subgraph induced by U^H) is connected and the variance σ_Δ^2 for the pairwise (canceling) noise is large enough. How large it should be depends on the topology of $\mathcal{G}^H$. Here, we restrict our attention on a practical instantiation of our general result where the graph $\mathcal{G}$ is obtained by a simple randomized procedure such that $\mathcal{G}^H$ will be well-connected with high probability, and prove a DP guarantee *for the whole process* (random graph generation followed by Gopa). The idea is to make each (honest) user select k other users uniformly at random among all users. Then, the edge $\{u, v\} \in E$ is created if u selected v or v selected u (or both).[3] Such

[3]Note that Gopa can be conveniently executed while constructing this graph.

graphs are known as *random k-out* or *random k-orientable* graphs (Bollobás, 2001; Fenner and Frieze, 1982; Yağan and Makowski, 2013). We have the following privacy guarantees.

Theorem 4.1 ([AB-Preprint7]). *Let $\epsilon, \delta \in (0,1)$ and let $\mathcal{G}$ be obtained by letting all (honest) users randomly choose $k \leq n$ neighbors. Let k and $\rho = n_H/n$ be such that $\rho n \geq 81$, $\rho k \geq 4\log(2\rho n/3\delta)$, $\rho k \geq 6\log(\rho n/3)$ and $\rho k \geq \frac{3}{2} + \frac{9}{4}\log(2e/\delta)$. If ϵ, δ, σ_η and σ_Δ satisfy*

$$\epsilon \geq \theta/2 + \theta^{1/2}, \tag{4.2}$$

$$(\epsilon - \theta/2)^2 \geq 2\log(2/\delta\sqrt{2\pi})\theta, \tag{4.3}$$

with

$$\theta = \frac{1}{n_H \sigma_\eta^2} + \frac{1}{\sigma_\Delta^2}\Big(\frac{1}{\lfloor (k-1)\rho/3 \rfloor - 1} + \frac{12 + 6\log(n_H)}{n_H}\Big),$$

then Gopa *is $(\epsilon, 3\delta)$-differentially private, i.e., $P(\hat{X} \mid X^A) \leq e^\epsilon P(\hat{X} \mid X^B) + 3\delta$.*

Sketch of proof. Our analysis starts by deriving an abstract (ϵ, δ)-DP result which holds for graphs $\mathcal{G}^H$ with an arbitrary topology. We model the knowledge of the adversary as a system of linear equations, which leads to a covariance structure for noisy values $\hat{X}^H$ that depend on the topology of $\mathcal{G}^H$. Requiring the privacy loss random variable to be smaller than ϵ with probability at least $1 - \delta$, we show that the privacy guarantees relate to the ability to "spread" the difference between the two neighboring datasets X^A and X^B over all vertices of $\mathcal{G}^H$. This result can then be concretely instantiated for specific topologies, which involves identifying a spanning tree of $\mathcal{G}^H$ with a large branching factor. While the optimal spanning tree is easy to construct for the worst-case of paths and for complete graphs, the case of random graphs is more involved and requires specific tools. We prove our result by leveraging and adapting results on embedding spanning trees in random graphs (Krivelevich, 2010). □

In order to get a constant ϵ, inspecting the term θ shows that the variance σ_η^2 of the independent noise must be of order $1/n_H$. This is in a sense optimal as it corresponds to the amount of noise required when averaging n_H values in the trusted curator model. It also matches the amount of noise needed when using secure aggregation with differential privacy in the presence of colluding users, where honest users need to add n/n_H more noise to compensate for collusion (Shi et al., 2011). In order to match the privacy-utility trade-off of the trusted curator setting, further inspection of the inequalities in Theorem 4.1 shows that k must be of order $\log(\rho n)/\rho)$ so that $\mathcal{G}^H$ is sufficiently connected despite dropouts and malicious users, while σ_Δ^2 needs to be of order $1/k\rho$.[4] Crucially,

[4]Recall that the pairwise noise cancels out, so it does not impact the utility of the final output. It only has a minor effect on the communication cost (the representation space of reals needs to be large enough to avoid overflows with high probability), and on the variance of the final result if some residual noise terms of dropout users are not rolled back (see Section 4.1.2).

	$\rho = 1$	$\rho = 0.5$
k-out graph (Theorem 4.1)	$\sigma_\Delta = 44.7$, $k = 105$	$\sigma_\Delta = 34.4$, $k = 203$
k-out graph (simulation)	$\sigma_\Delta = 34.7$, $k = 20$	$\sigma_\Delta = 28.4$, $k = 40$

Table 4.1: Values of σ_Δ and k needed to ensure (ϵ, δ)-DP with trusted curator utility for $n = 10000$, $\epsilon = 0.1$, $\delta' = 1/n_H^2$, $\delta = 10\delta'$ with a random k-out graph.

each user needs to exchange with only $2k = O(\log n)$ peers in expectation, which is much more scalable than the $O(n)$ communication cost per user of classic secure aggregation approaches (Bonawitz et al., 2017).

Scaling the noise in practice. Using Theorem 4.1, we can precisely quantify the amount of independent and pairwise noise needed to achieve a desired privacy guarantee, as illustrated by the following corollary.

Corollary 4.1. *Let $\epsilon, \delta' \in (0,1)$, and $\sigma_\eta^2 = c^2/n_H\epsilon^2$, where $c^2 > 2\log(1.25/\delta')$. Given some $\kappa > 0$, let $\sigma_\Delta^2 = \kappa\sigma_\eta^2 n_H(\frac{1}{\lfloor(k-1)\rho/3\rfloor - 1} + (12 + 6\log(n_H))/n_H)$ with k satisfying the conditions of Theorem 4.1. Then* Gopa *is (ϵ, δ)-DP with $\delta \geq 3.75(\delta'/1.25)^{\kappa/\kappa+1}$.*

In Corollary 4.1, σ_η^2 is set such that after all noisy values are aggregated, the variance of the residual noise matches that required by the Gaussian mechanism to achieve (ϵ, δ')-DP for an average of n_H values in the *centralized* setting. The privacy-utility trade-off achieved by Gopa is thus the same as in the trusted curator model up to a small constant in δ, as long as the pairwise variance σ_Δ^2 is large enough. As expected, we see that as $\sigma_\Delta^2 \to +\infty$ (that is, as $\kappa \to +\infty$), we have $\delta \to \delta'$. Given the desired $\delta \geq \delta'$, we can use Corollary 4.1 to determine a value for σ_Δ^2 that is sufficient for Gopa to achieve (ϵ, δ)-DP. Table 4.1 shows a numerical illustration with δ only a factor 10 larger than δ'. We report the values of σ_Δ and k given by Theorem 4.1, as well as smaller (yet admissible) values obtained by numerical simulation (see **[AB-Preprint7]** for details). Although the conditions of Theorem 4.1 are a bit conservative (constants can likely be improved), they still lead to practical values. Note that in practice, one often does not know in advance the exact proportion ρ of users who are honest and will not drop out, so a lower bound can be used instead.

Remark 4.1. *Our privacy guarantees protect against an adversary that consists of colluding malicious users. To simultaneously protect against each single honest-but-curious user (who knows his own independent noise term), we can simply replace n_H by $n'_H = n_H - 1$ in our results. This introduces a factor $n_H/(n_H - 1)$ in the variance, which is negligible for large n_H.*

4.1.4 Correctness Against Malicious Users

While the privacy guarantees of Section 4.1.3 hold regardless of the behavior of malicious users, the utility guarantees discussed in Section 4.1.2 are not valid if malicious users tamper with the protocol. In this section, we use existing cryptographic primitives to ensure the correctness of the computations while preserving privacy. We only give a high-level description of our approach and refer to **[AB-Preprint7]** for details.

While it is impossible to force a user to give the "right" input to the algorithm, this also holds in the centralized setting. Our goal is thus to guarantee that given the input vector X, we can identify malicious behavior and generate a truthfully computed $\hat{X}^{avg}$ which excludes any faulty contributions. Concretely, users will be able to prove the following properties:

$$\hat{x}_u = x_u + \sum_{v \in N(u)} \Delta_{u,v} + \eta_u, \qquad \forall u \in U, \quad (4.4)$$

$$\Delta_{u,v} = -\Delta_{v,u}, \qquad \forall \{u,v\} \in E, \quad (4.5)$$

$$\eta_u \sim \mathcal{N}(0, \sigma_\eta^2), \qquad \forall u \in U, \quad (4.6)$$

$$x_u \in [0,1], \qquad \forall u \in U. \quad (4.7)$$

It is easy to see that the correctness of the computation is guaranteed if Properties (4.4)-(4.7) are satisfied. Note that, as long as they are self-canceling and not excessively large (avoiding overflows and additional costs if a user drops out), we do not need to ensure that pairwise noise terms $\Delta_{u,v}$ have been drawn from the prescribed distribution, as these terms do not influence the final result and only those involving honest users affect the privacy guarantees of Section 4.1.3. In contrast, Properties (4.6)-(4.7) are necessary to prevent a malicious user from biasing the outcome of the computation. Indeed, (4.6) ensures that the independent noise is generated correctly, while (4.7) ensures that input values are in the allowed range.

Tools for verifying computations. Our approach consists in publishing an encrypted log of the computation using *cryptographic commitments* and proving that it is performed correctly without revealing any additional information using *zero knowledge proofs*. Moreover, signing the messages adds non-repudiation. These techniques are popular in a number of applications such as privacy-friendly auditable financial systems like Zcash and Findora. We here give a brief overview of the functionality of these tools and their role in verifying Properties (4.4)-(4.7).

Commitments, first introduced by Blum (1983), allow users to commit to chosen values while keeping them hidden from others. After the commitment is performed, the committer cannot change its value, but can later reveal it or prove properties of it. For our protocol we use an optimized version (Franck and Großschädl, 2017) of the Pedersen commitment scheme (Pedersen, 1991). A commitment is obtained by transforming

the hidden statement using a hard-to-invert injective function Com, so that the recipient cannot know the original statement but can be sure that the committer cannot change the statement and still obtain the same commitment. Pedersen commitments additionally satisfy the *homomorphic property*, meaning that $Com(v_1 + v_2) = Com(v_1) + Com(v_2)$ for all values v_1, v_2. This property facilitates the verification of summations without revealing them, in our case Properties (4.4) and (4.5).

To verify other properties than the additive ones, we use another family of cryptographic operations known as Zero Knowledge Proofs (ZKPs). Informally speaking, ZKPs allow a user (prover) to effectively prove a true statement (completeness), but also allow the other user (verifier) to discover with high probability if a cheating prover is trying to prove a statement which is not true (soundness), in such a way that by performing the proof no information other than the proven statement is revealed (zero knowledge). Here, we use classic proof techniques (see e.g., Schnorr, 1991; Chaum and Pedersen, 1993; Fujisaki and Okamoto, 1997) as building blocks. In particular, these building blocks support the verification of (4.4) and (4.5), and constitute the bulk of the verification of (4.6) and (4.7). The only assumption needed to ensure the validity of the cryptographic building blocks on which we rely is the Discrete Logarithm Assumption (DLA).

Verification protocol. In a setup phase, users agree on a commitment function and generate random seeds. Then, while executing GOPA, users publish commitments of x_u and η_u for all $u \in U$ and of $\Delta_{u,v}$ for all $\{u, v\} \in E$. Due to the homomorphic property of the Pedersen commitments, everyone can then verify the summation relations (4.4) and (4.5). In a second phase, ZKPs are performed to prove the remaining parts of the verification. We implement the publication of commitments using a public bulletin board so that any party can verify the validity of the protocol, avoiding the need for a trusted verification entity. Users sign their messages so they cannot deny them. Since the basic cryptographic operations are integer-based, we use a fixed precision scheme, which is efficient as our computations are mostly additive. The following result summarizes the security guarantees.

Theorem 4.2 ([**AB-Preprint7**]). *Under the DLA, a user $u \in U$ that passes the verification procedure proves that $\hat{x}_u$ was computed correctly. Additionally, u does not reveal any additional information about x_u by running the verification, even if DLA does not hold.*

4.2 Privacy Amplification by Decentralization

The previous approach was targeted to decentralized learning with an untrusted aggregator. In this section, we focus on fully decentralized algorithms and show that they can naturally amplify privacy guarantees through the introduction of novel relaxation of

local differential privacy. This relaxation, that we call network DP, is introduced in Section 4.2.1. Section 4.2.2 presents the decentralized model of computation that we study, in which a token performs a walk on the network graph and is updated sequentially by the user who receives it. Section 4.2.3 and Section 4.2.4 introduce simple algorithms for the ring and complete topologies and show that their privacy-utility trade-offs under network DP significantly improve upon what is achievable under LDP (sometimes matching the utility of the trusted curator model). Finally, Section 4.2.5 illustrates the improved utility of our approach for decentralized training with stochastic gradient descent.

4.2.1 Network Differential Privacy

We consider a set $[\![n]\!]$ of users, which are assumed to be honest-but-curious (i.e., they truthfully follow the protocol). Each user u holds a private dataset $\mathcal{D}_u$, and we denote by $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_n$ the union of all user datasets. In this work, we consider *user-level* DP (see Remark 1.1): two datasets $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_u \cup \cdots \cup \mathcal{D}_n$ and $\mathcal{D}' = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}'_u \cup \cdots \cup \mathcal{D}_n$ are neighboring if they differ only on user u's data, which we denote by $\mathcal{D} \sim_u \mathcal{D}'$. This relation is weaker than the one used in record-level DP and thus provides stronger privacy guarantees.

We consider a fully decentralized setting, in which users are nodes in a network graph $\mathcal{G} = ([\![n]\!], E)$ and an edge $(u, v) \in E$ indicates that user u can send messages to user v. Here, the graph may be directed or undirected, and could in principle change over time although we will restrict our attention to fixed topologies. For the purpose of quantifying privacy guarantees, a decentralized algorithm $\mathcal{A}$ will be viewed as a (randomized) mapping which takes as input a dataset $\mathcal{D}$ and outputs the transcript of all messages exchanged between users over the network. We denote the (random) output in an abstract manner by $\mathcal{A}(\mathcal{D}) = ((u, m, v) :$ user u sent message with content m to user $v)$.

We introduce a new relaxation of LDP whose key idea is to consider that *a given user does not have access to the full transcript $\mathcal{A}(\mathcal{D})$ but only to the messages he/she is involved in*, which can be enforced by the use of secure communication channels. We denote the corresponding view of a user u by

$$\mathcal{O}_u(\mathcal{A}(\mathcal{D})) = ((v, m, v') \in \mathcal{A}(\mathcal{D}) : v = u \text{ or } v' = u). \tag{4.8}$$

Definition 4.1 (Network Differential Privacy [**AB-Preprint6**]). *An algorithm $\mathcal{A}$ satisfies (ϵ, δ)-network DP if for all pairs of distinct users $u, v \in [\![n]\!]$ and all pairs of neighboring datasets $\mathcal{D} \sim_u \mathcal{D}'$, we have:*

$$\Pr(\mathcal{O}_v(\mathcal{A}(\mathcal{D}))) \leq e^{\epsilon} \Pr(\mathcal{O}_v(\mathcal{A}(\mathcal{D}'))) + \delta. \tag{4.9}$$

Network DP essentially requires that for any two users u and v, the information gathered by user v during the execution of $\mathcal{A}$ should not depend too much on user u's data. Network DP can be thought of as analyzing the composition of the operator $\mathcal{O}_v$ with

$\mathcal{A}$. We will show that for some algorithms, $\mathcal{O}_v \circ \mathcal{A}$ is more private than $\mathcal{A}$ (i.e., applying $\mathcal{O}_v$ *amplifies* the privacy guarantees of $\mathcal{A}$). Note that if $\mathcal{O}_v$ is the identity map (i.e., if each user is able to observe all messages), then (4.9) boils down to local DP.

Definition 4.1 can be naturally extended to account for potential *collusions* between users. We consider an upper bound c on the number of users that can possibly collude. In this setting, we would like to be private with respect to the aggregated information $\mathcal{O}_V = \cup_{v \in V} \mathcal{O}_v$ acquired by any possible subset V of c users, as captured by the following generalization of Definition 4.1.

Definition 4.2 (Network DP with collusions [**AB-Preprint6**]). *An algorithm $\mathcal{A}$ is (c, ϵ, δ)-network DP if for any user $u \in [\![n]\!]$, all subsets $V \subset [\![n]\!]$ such that $|V| \leq c$ and $u \notin V$, and all pairs of neighboring datasets $D \sim_u D'$, we have:*

$$\Pr(\mathcal{O}_V(\mathcal{A}(\mathcal{D})) \leq e^{\epsilon} \Pr(\mathcal{O}_V(\mathcal{A}(\mathcal{D}')) + \delta. \tag{4.10}$$

4.2.2 Decentralized Computation Model

We will use network DP to study the privacy guarantees of decentralized algorithms that perform computations via sequential updates to a *token* τ walking through the nodes by following the edges of the graph $\mathcal{G}$. At each step, the token τ resides at some user u and is updated by

$$\tau \leftarrow \tau + x_u^k, \quad \text{with } x_u^k = g^k(\tau; D_u), \tag{4.11}$$

where $x_u^k = g^k(\tau; D_u)$ denotes the contribution of user u. The notation highlights the fact that this contribution may depend on the current value τ of the token as well as on the number of times k that the token visited u so far. The token τ is then sent to another user v such that $(u, v) \in E$.

Provided that the walk follows some properties (e.g., corresponds to a deterministic cycle or a random walk that is suitably ergodic), this model of computation allows to optimize sums of local objective functions of the form (1.2) using (stochastic) gradient descent (Ram et al., 2009; Johansson et al., 2009; Mao et al., 2020; Ayache and El Rouayheb, 2021).[5] In this case, the token τ holds the model parameters and x_u^k is a (stochastic) gradient of the local objective of user u evaluated at τ. Such decentralized algorithms can also be used to compute summaries of the users' data, for instance any commutative and associative operation like sums/averages and discrete histograms. In these cases, the contributions of a given user may correspond to different values acquired over time.

4.2.3 Privacy Amplification for Walking on a Ring

In this section, we analyze a simple case where the graph is a directed ring, i.e., $E = \{(u, u+1)\}_{u=1}^{n-1} \cup \{(n, 1)\}$. The token starts at user 1 and goes through the ring K

[5] These algorithms are sometimes referred to as *incremental gradient methods*.

Algorithm 4.2 Private real summation on the ring.

Public Parameters: Ring graph on $[\![n]\!]$; Noise scale $\sigma_{loc} \in \mathbb{R}^+$; Number of rounds K
Input: Contributions $x_u^1, \ldots, x_u^K \in \mathbb{R}$ for each user $u \in [\![n]\!]$

```
τ ← 0; a ← 0
for k = 1 to K do
  for u = 1 to n do
    if a = 0 then
      τ ← τ + Perturb(x_u^k; σ_loc); a ← n − 2
    else
      τ ← τ + x_u^k; a ← a − 1
return τ
```

times. The ring (i.e., ordering of the nodes) is assumed to be public.

Real summation. We first consider the task of estimating the sum $\bar{x} = \sum_{u=1}^{n} \sum_{k=1}^{K} x_u^k$ where the x's are bounded real numbers and x_u^k represents the contribution of user u at round k. For this problem, the standard approach in local differential privacy (LDP) is to add random noise to each single contribution before releasing it. For generality, we consider an abstract mechanism $\mathrm{Perturb}(x; \sigma)$ which adds centered noise with standard deviation σ to the contribution x (e.g., the Gaussian or Laplace mechanism). Let σ_{loc} be the standard deviation of the noise required so that $\mathrm{Perturb}(\cdot; \sigma_{loc})$ satisfies (ϵ, δ)-LDP.

Consider now the simple decentralized protocol in Algorithm 4.2, where noise with the same standard deviation σ_{loc} is added *only once every $n-1$ hops of the token*. By leveraging the fact that the view of each user u is restricted to the values taken by the token at each of its K visits to u, combined with advanced composition (Dwork et al., 2010b), we have the following result.

Theorem 4.3 ([**AB-Preprint6**]). *Let $\epsilon, \delta > 0$. Algorithm 4.2 outputs an unbiased estimate of $\bar{x}$ with standard deviation $\sqrt{\lfloor Kn/(n-1) \rfloor}\sigma_{loc}$, and satisfies $(\sqrt{2K\ln(1/\delta')}\epsilon + K\epsilon(e^{\epsilon} - 1), K\delta + \delta')$-network DP for any $\delta' > 0$.*

To match the same privacy guarantees, LDP incurs a standard deviation of $\sqrt{Kn}\sigma_{loc}$. Therefore, Algorithm 4.2 provides an $O(1/\sqrt{n})$ reduction in error or, equivalently, an $O(1/\sqrt{n})$ gain in ϵ. In fact, Algorithm 4.2 achieves the same privacy-utility trade-off as a *trusted central aggregator* that would iteratively aggregate the raw contributions of all users at each round k and perturb the result before sending it back to the users, as done in federated learning algorithms with a trusted aggregator [**AB-Journal2**].

Remark 4.2. *We can design variants of Algorithm 4.2 in which noise addition is distributed across users. Using the Gaussian mechanism, each user can add noise with std. dev. $\sigma'_{loc} = \sigma_{loc}/\sqrt{n}$, except for the very first contribution which requires std. dev. σ_{loc} to properly hide the contributions*

Algorithm 4.3 Private discrete histogram computation on the ring.

Public Parameters: Ring graph on $[\![n]\!]$; Noise parameter $\beta \in [0,1]$; Number of rounds K
Input: Contributions $x_u^1, \ldots, x_u^K \in [\![d]\!]$ for each user $u \in [\![n]\!]$
  Initialize $\tau \in \mathbb{N}^d$ with βn random elements
  for $k = 1$ to K **do**
    for $u = 1$ to n **do**
      $y_u^k \leftarrow RR_\beta(x_u^k)$
      $\tau[y_u^k] \leftarrow \tau[y_u^k] + 1$
  for $i = 0$ to $d-1$ **do**
    $\tau[i] \leftarrow \frac{\tau[i] - \beta/d}{1-\beta}$
  return τ

of users in the first cycle. The total added noise has std. dev. $\sqrt{\lfloor Kn/(n-1) \rfloor + 1}\sigma_{loc}$*, leading to same utility as Algorithm 4.2 (up to a constant factor that is negligible when K is large).*

Discrete histogram computation. We now turn to histogram computation over a discrete domain $[\![d]\!] = \{1, \ldots, d\}$. The goal is to compute $h \in \mathbb{N}^d$ s.t. $h_i = \sum_{u=1}^n \sum_{k=1}^K \mathbb{I}[x_u^k = i]$, where $x_u^k \in [\![d]\!]$ and $\mathbb{I}[\cdot]$ is the indicator function. A classic approach in LDP is based on d-ary randomized response (Kairouz et al., 2014), where each user submits its true value with probability $1-\beta$ and a uniformly random value with probability β.[6] We denote this primitive by $RR_\beta : [\![d]\!] \to [\![d]\!]$.

In our setting with a ring network, we propose Algorithm 4.3, where each contribution of a user is randomized using RR_β before being added to the token $\tau \in \mathbb{N}^d$. Additionally, τ is initialized with enough random elements to hide the first contributions. Note that at each step, the token contains a partial histogram equivalent to a shuffling of the contributions added so far, allowing us to leverage results on *privacy amplification by shuffling* (Erlingsson et al., 2019; Balle et al., 2019; Feldman et al., 2020). In particular, we can prove the following utility and privacy guarantees for Algorithm 4.3.

Theorem 4.4 ([AB-Preprint6]). *Let* $\epsilon < \frac{1}{2}$*,* $\delta \in (0, \frac{1}{100})$*, and* $n > 1000$*. Let* $\beta = d/(\exp(12\epsilon\sqrt{\log(1/\delta)/n}) + d - 1)$*. Algorithm 4.3 outputs an unbiased estimate of the histogram with* $\beta n(K+1)$ *expected random responses. Furthermore, it satisfies* $(\sqrt{2K\ln(1/\delta')}\epsilon + K\epsilon(e^\epsilon - 1), K\delta + \delta')$*-network DP for any* $\delta' > 0$*.*

Achieving the same privacy guarantees in LDP would require β to be constant in n, hence $\sqrt{n}$ times more random responses. Equivalently, if we fix utility (i.e., β), Theorem 4.4 shows that Algorithm 4.3 again provides a privacy gain of $\frac{1}{n}\sqrt{n/\ln(1/\delta)} = O(1/\sqrt{n})$ compared to LDP.

[6] Recall that we used this local randomizer in Chapter 3 to privately compute U-statistics in the local model, see Algorithm 3.3.

Remark 4.3. *For simplicity of presentation, Theorem 4.4 relies on the amplification by shuffling result of Erlingsson et al. (2019) which has a simple closed-form. A tighter and more general result (with milder restrictions on the values of n, ϵ and δ) can be readily obtained by using the results of Balle et al. (2019) and Feldman et al. (2020).*

Remark 4.4. *Algorithm 4.2 (real summation) could also be used to perform histogram computation. However, for domains of large cardinality d (e.g., $d \gg n$), Algorithm 4.3 requires fewer random numbers and maintains a sparse (thus more compact) representation of the histogram.*

Discussion. We have seen that decentralized computation over a ring provides a simple way to achieve utility similar to a trusted aggregator thanks to the sequential communication that hides the contribution of the previous users in a summary. We stress the fact that this is achieved without relying on a central server (only local communications) or resorting to costly multi-party computation protocols (only two secure communication channels per user are needed). We observe that the ring topology is often used in practical deployments and theoretical analysis of (non-private) decentralized algorithms (Lian et al., 2017; Tang et al., 2018; Koloskova et al., 2020; Neglia et al., 2020; Marfoq et al., 2020), owing to its simplicity and good empirical performance. Finally, we note an interesting connection between the specific case of network DP over a ring topology and the pan-privacy model for streaming algorithms (Dwork et al., 2010a), see **[AB-Preprint6]**.

Despite these important advantages, the use of a fixed ring topology has some limitations. First, our algorithms are not robust to collusions: in particular, if two users collude and share their view, Algorithm 4.2 does not satisfy DP. While this can be mitigated by distributing the noise addition across users (Remark 4.2), a node placed between two colluding nodes would get largely degraded privacy guarantees. A similar reasoning holds for Algorithm 4.3. Second, a fixed ring topology is not well suited to extensions to gradient descent, for which we would like to leverage privacy amplification by iteration (Feldman et al., 2018). In this amplification scheme, the privacy guarantee for a given user (data point) grows with the number of gradient steps that come after it. In a fixed ring, the privacy of a user u with respect to another user v would thus depend on their relative positions in the ring (e.g., there would be no privacy amplification when v is the user who comes immediately after u). These limitations motivate us to consider random walks on a complete graph.

4.2.4 Privacy Amplification for Walking on a Complete Graph

In this section, we consider the case of a random walk on the complete graph. In other words, at each step, the token is sent to a user chosen uniformly at random among V. We consider random walks of fixed length $T > 0$, hence the number of times a given user contributes is itself random.

Algorithm 4.4 Private real summation on a complete graph.

Public Parameters: Noise scale $\sigma_{loc} \in \mathbb{R}^+$; Number of steps T
Input: Contributions $x_u^1, \ldots, x_u^T \in \mathbb{R}$ for each user $u \in [\![n]\!]$
  $\tau \leftarrow 0, k_1 \leftarrow 0, \ldots, k_n \leftarrow 0$
  for $t = 1$ to T **do**
    Draw $u \sim \mathcal{U}(1, \ldots, n)$
    $k_u \leftarrow k_u + 1$
    $\tau \leftarrow \tau + \text{Perturb}(x_u^{k_u}; \sigma_{loc})$
  return τ

Remark 4.5. *For simplicity, in this section we slightly depart from the notion of view defined in (4.8). Specifically, we assume that the path taken by the token is fully hidden from a given user, i.e., the user does not know the identity of the sender (resp. receiver) of messages that he/she receives (resp. sends). We discuss how this assumption can be lifted at the end of the section.*

Real summation. For real summation, we consider the simple and natural protocol shown in Algorithm 4.4: a user u receiving the token τ for the k-th time updates it with $\tau \leftarrow \tau + \text{Perturb}(x_u^k; \sigma_{loc})$. As in Section 4.2.3, σ_{loc} is set such that $\text{Perturb}(\cdot; \sigma_{loc})$ satisfies (ϵ, δ)-LDP, and thus implicitly depends on ϵ and δ. The next theorem gives network DP guarantees, which rely on *the intermediate aggregations of values between two visits of the token to a given user* and *the secrecy of the path taken by the token*. For clarity, we state below an asymptotic result to give the main order of magnitude, but stress the fact that it is derived from a *non-asymptotic* (albeit more complex) formula given in [**AB-Preprint6**].

Theorem 4.5 ([**AB-Preprint6**]). *Let $\epsilon < 1$ and $\delta > 0$. Algorithm 4.4 outputs an unbiased estimate of the sum of T contributions with standard deviation $\sqrt{T}\sigma_{loc}$. Furthermore, for large enough n and $T = \Omega(n)$, it satisfies $(\epsilon', N_v\delta + \delta' + \hat{\delta})$-network DP for all $\delta', \hat{\delta} > 0$ with*

$$\epsilon' = O\Big(\sqrt{N_v \ln(1/\delta')}\epsilon/n^{1/3}\Big), \quad \textit{where } N_v = \frac{T}{n} + \sqrt{\frac{3}{n} T \ln(1/\hat{\delta})}. \tag{4.12}$$

Sketch of proof. We fix a user v and quantify how much information about the private data of another user u is leaked to v from the visits of the token. The number of contributions from u follows a binomial law $\mathcal{B}(T, 1/n)$: we can bound it by N_u with prob. $1 - \hat{\delta}$ using Chernoff. Then, for a contribution of u at time t, it is sufficient to consider the cycle formed by the random walk between the two successive passages in v containing t. We distinguish whether the contribution is part of a "small cycle" (i.e., aggregated with less than $c^2 n^{2/3}$ between two visits to v), or in a larger cycle. For small cycles, we can use amplification by subsampling Balle et al., 2018 thanks to the secrecy of the cycle. For larger cycles, the contribution of u is aggregated with at least $c^2 n^{2/3}$ other contributions, hence the privacy loss of this contribution towards v is bounded by $O(\epsilon/n^{1/3})$. The total privacy loss across the N_u contributions follows from advanced composition. □

Algorithm 4.5 Private SGD on a complete graph.

Public Parameters: Privacy parameters $\epsilon, \delta > 0$; Number of steps T; Step sizes $(\gamma(t))_{t=1}^{T}$
Input: Dataset $\mathcal{D}_u$ for each user $u \in [\![n]\!]$
1: Initialize $\tau \in \mathcal{C}$
2: **for** $t = 1$ to T **do**
3: Draw $u \sim \mathcal{U}(1, \ldots, n)$
4: $\eta = [\eta_1, \ldots, \eta_p]$, with $\eta_i \sim \mathcal{N}\big(0, \frac{8L^2 \ln(1.25/\delta)}{\epsilon^2}\big)$
5: $\tau \leftarrow \Pi_{\mathcal{C}}\big(\tau - \gamma(t)[\nabla_\tau F_u(\tau; \mathcal{D}_u) + \eta]\big)$
6: **return** τ

The same algorithm analyzed under LDP yields $\epsilon' = O(\sqrt{N_v \ln(1/\delta')}\epsilon)$, which is optimal for averaging N_v contributions per user in the local model. Theorem 4.5 thus shows that network DP asymptotically provides a privacy amplification of $O(1/n^{1/3})$ over LDP. While this theoretical gain is not as strong as the one obtained for the fixed ring topology, we show in **[AB-Preprint6]** that our non-asymptotic formula improves upon local DP as soon as $n > 100$, and that the gains are significantly stronger in practice than what our theoretical results guarantee.

Remark 4.6 (Extension to discrete histogram computation)**.** *We can obtain a similar result for histograms by bounding the privacy loss incurred by larger cycles in the proof of Theorem 4.5 using amplification by shuffling (Erlingsson et al., 2019; Balle et al., 2019; Feldman et al., 2020), similar to what we did for the ring in Section 4.2.3.*

Optimization with stochastic gradient descent (SGD). We now turn to decentralized learning with SGD. Let $\mathcal{C} \subseteq \mathbb{R}^p$ be a convex set and $F_1(\cdot; \mathcal{D}_1), \ldots, F_n(\cdot; \mathcal{D}_n)$ be a set of convex L-Lipschitz and s-smooth local objective functions over $\mathcal{C}$ (see Appendix B for a reminder on these notions). We denote by $\Pi_{\mathcal{C}}(\theta) = \arg\min_{\theta' \in \mathcal{C}} \|\theta - \theta'\|$ the Euclidean projection onto the set $\mathcal{C}$. We aim to privately solve the following optimization problem:

$$\theta^* \in \arg\min_{\theta \in \mathcal{C}} \Big\{ F(\theta) = \frac{1}{n} \sum_{u=1}^{n} F_u(\theta; \mathcal{D}_u) \Big\}. \tag{4.13}$$

Note that (4.13) corresponds to the classic decentralized learning objective (1.2), where we have ignored the weights for simplicity.

To privately approximate θ^*, we propose Algorithm 4.5. Here, the token $\tau \in \mathcal{C}$ represents the current iterate. At each step, the user u with the token performs a projected noisy gradient step and sends the updated token to a random user. We rely on the Gaussian mechanism to ensure that the noisy version of the gradient $\nabla_\tau F_u(\tau; \mathcal{D}_u) + \eta$ satisfies (ϵ, δ)-LDP: the variance σ^2 of the noise η in line 4 of Algorithm 4.5 follows from the fact that gradients of L-Lipschitz functions have ℓ_2-sensitivity bounded by $2L$ (this is a consequence of Lemma B.1). We have the following network DP guarantee.

Theorem 4.6. *Let $\epsilon < 1$, $\delta < 1/2$. Algorithm 4.5 with step size $\eta \leq 2/s$ achieves $(\epsilon', \delta + \hat{\delta})$-network DP for all $\hat{\delta} > 0$ with*

$$\epsilon' = \sqrt{2q \ln(1/\delta)}\epsilon / \sqrt{\ln(1.25/\delta)}, \tag{4.14}$$

where $N_u = \frac{T}{n} + \sqrt{\frac{3}{n} T \ln(1/\hat{\delta})}$ and $q = \max\left(\frac{2N_u \ln n}{n}, 2\ln(1/\delta)\right)$.

Sketch of proof. The proof tracks the evolution of the privacy loss using Rényi Differential Privacy (RDP) (Mironov, 2017) and leverages amplification by iteration (Feldman et al., 2018) in a novel decentralized context. We give here a brief sketch. Let us fix two users u and v and bound the privacy leakage of u from the point of view of v. The number of contributions of user u is bounded by N_u as in Theorem 4.5. We then compute the network RDP guarantee for a fixed contribution of u at time t. Crucially, it is sufficient to take into account the first time that v receives the token at a step $t' > t$. Privacy amplification by iteration tells us that the larger t', the less is learned by v about the contribution of u. Note that t' follows a geometric law of parameter $1/n$. Using the weak convexity of the Rényi divergence proved in (Feldman et al., 2018, Lemma 25 therein), we can bound the Rényi divergence $D_\alpha(Y_v || Y'_v)$ between two random executions Y_v and Y'_v stopping at v and differing only in the contribution of u by the expected divergence over the geometric distribution. Combining with amplification by iteration eventually gives us $D_\alpha(Y_v || Y'_v) \leq 4\alpha L^2 \ln n / \sigma^2 n$. We conclude by applying the composition property of RDP over the N_u contributions of u and converting the RDP guarantee into (ϵ, δ)-DP. □

Algorithm 4.5 is also a natural approach to private SGD in the local model, and achieves $\epsilon' = O(\sqrt{N_u \ln(1/\delta')}\epsilon)$ under LDP. Thus, for $T = \Omega(n^2 \sqrt{\ln(1/\delta)} / \ln n)$ iterations, Theorem 4.6 gives a privacy amplification of $O(\ln n / \sqrt{n})$ compared to LDP. Measuring utility as the amount of noise added to the gradients, the privacy-utility trade-off of Algorithm 4.5 in network DP is thus nearly the same (up to a log factor) as that of private SGD in the trusted curator model![7] For smaller T, the amplification is still much stronger than suggested by the closed-form in (4.14): we can numerically find a smaller ϵ' that satisfies the conditions required in the proof, see **[AB-Preprint6]** for details.

We note that we can easily obtain utility guarantees for Algorithm 4.5 in terms of optimization error. Indeed, the token performs a random walk on a complete graph so the algorithm performs the same steps as a *centralized* (noisy) SGD algorithm. We can for instance rely on a classic result by Shamir and Zhang (2013, Theorem 2 therein) which shows that SGD-type algorithms applied to a convex function and bounded convex domain converge in $O(1/\sqrt{T})$ as long as gradients are unbiased with bounded variance.

[7] Incidentally, the analysis of centralized private SGD (Bassily et al., 2014) also sets the number of iterations to be of order n^2.

Proposition 4.1. *Let the diameter of $\mathcal{C}$ be bounded by D. Let $G^2 = L^2 + \frac{8pL^2 \ln(1.25/\delta)}{\epsilon^2}$, and $\tau \in \mathcal{C}$ be the output of Algorithm 4.5 with step size $\gamma(t) = D/G\sqrt{t}$. Then we have:*

$$\mathbb{E}[F(\tau) - F(\theta^*)] \leq 2DG(2 + \log T)/\sqrt{T}.$$

A consequence of Proposition 4.1 and Theorem 4.6 is that for fixed privacy budget and sufficiently large T, the expected error of Algorithm 4.5 is $O(\ln n/\sqrt{n})$ smaller under network DP than under LDP.

Discussion. An advantage of considering a random walk over a complete graph is that our approach is naturally robust to the presence of a (constant) number of colluding users. Indeed, when c users collude, they can be seen as a unique node in the graph with a transition probability of $\frac{c}{n}$ instead of $\frac{1}{n}$. We can then easily adapt the proofs above using the fact that the total number of visits to colluding users follows $\mathcal{B}(T, c/n)$ and that the size of a cycle between two colluding users follows a geometric law of parameter $1 - c/n$. Hence, we obtain the same guarantees under Definition 4.2 as for the case with n/c non-colluding users under Definition 4.1. Interestingly, these privacy guarantees hold even if colluding users bias their choice of the next user instead of choosing it uniformly.

The assumption that users do not know the identity of the previous sender and the next receiver (see Remark 4.5) may seem quite strong. It is however possible to lift this assumption by bounding (with high probability) the number of times a contribution of a given user u is directly observed by a given user v and accounting for the resulting privacy loss separately. When $T = \Omega(n^2)$, this term is negligible and the results of Theorems 4.5-4.6 still hold. In practical implementations, we can enforce a deterministic bound on the number of times any edge (u, v) is used, e.g., by contributing only noise along (u, v) after it has been used too many times. We refer to **[AB-Preprint6]** for details.

4.2.5 Experiments on Private Stochastic Gradient Descent

We present some numerical experiments that illustrate the practical significance of our privacy amplification results in the complete graph setting (Section 4.2.4). We focus on the task of training a logistic regression model in the decentralized setting. Logistic regression corresponds to solving (4.13) with $\mathcal{C} = \mathbb{R}^p$ and $F_u(\theta; \mathcal{D}_u) = \frac{1}{|\mathcal{D}_u|} \sum_{(x,y) \in \mathcal{D}_u} \ln(1 + \exp(-y\theta^\top x))$ where $x \in \mathbb{R}^p$ and $y \in \{-1, 1\}$. We use a binarized version of UCI Housing dataset.[8] We standardize the features and further normalize each data point x to have unit L2 norm so that the logistic loss is 1-Lipschitz for any (x, y). We split the dataset uniformly at random into a training set (80%) and a test set, and further split the training set across $n = 2000$ users, resulting in each user u having a local dataset $\mathcal{D}_u$ of size 8.

We compare three private SGD approaches which are all based on gradient perturbation with the Gaussian mechanism. *Centralized DP-SGD* is the centralized version of

[8] https://www.openml.org/d/823

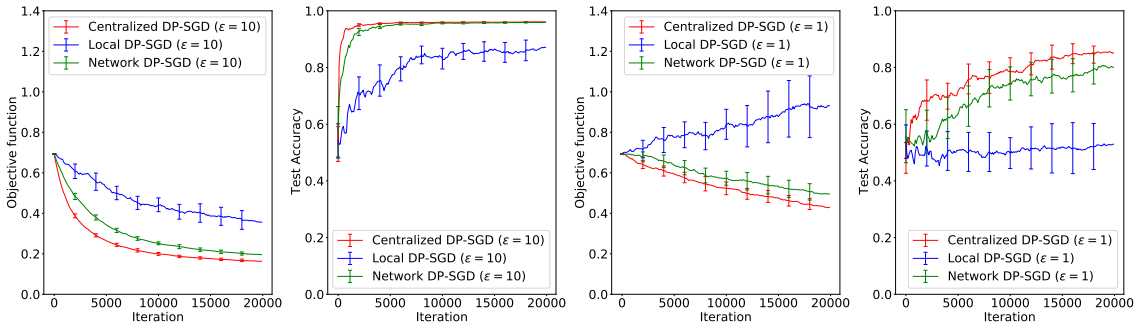


Figure 4.1: Comparing three settings for SGD with gradient perturbation. We plot the mean/std. dev. of the objective function and test accuracy over 20 runs.

differentially private SGD (*Centralized DP-SGD*) introduced by Song et al. (2013) and Bassily et al. (2014), which assumes the presence of a trusted curator/aggregator. *Local DP-SGD* corresponds to Algorithm 4.5 with noise calibrated for the LDP setting. Finally, *Network DP-SGD* is Algorithm 4.5 with noise calibrated according to network DP (see Theorem 4.6). To make the comparison as fair as possible, all approaches use the full dataset $\mathcal{D}_u$ of a randomly chosen user u as the mini-batch at each step.

Given the overall privacy budget (ϵ, δ), each of the three methods leads to a different choice for σ that parametrizes the level of noise added to each gradient. In our experiments, we consider $\epsilon = 10$ (low privacy) and $\epsilon = 1$ (stronger privacy), and fix $\delta = 10^{-6}$. Recall that we consider user-level DP. Note that due to composition, more iterations increase the per-iteration level of noise needed to achieve a fixed DP guarantee. As the number of contributions of a given user is random, we upper bound it in advance with a tighter bound than used in our theorems, namely cT/n where c is a parameter to tune. If a user is asked to participate more times than budgeted, it simply forwards the token to another user without adding any contribution. In the case of Network DP-SGD, the user still adds noise as the privacy guarantees of others rely on it. Note that the best regime for network DP is when the number of contributions of a user is roughly equal to n, see Theorem 4.6. In our experiments, we are not in this regime but the privacy amplification effect is stronger than the closed form of the theorem. In practice, we compute numerically the smallest σ needed to fulfill the conditions of the proof.

Figure 4.1 shows results for $T = 20,000$, where the step size η was tuned separately for each approach in the interval $[10^{-4}, 2]$. We see that Network DP-SGD nearly matches the privacy-utility trade-off of Centralized DP-SGD for both $\epsilon = 1$ and $\epsilon = 10$ without relying on a trusted curator. Network DP-SGD also clearly outperforms Local DP-SGD, which actually diverges for $\epsilon = 1$. These empirical results are consistent with our theory and show that Network DP-SGD significantly amplifies privacy guarantees compared to local DP-SGD even when the number of iterations T is much smaller than $O(n^2/\ln n)$, a regime which is of much practical importance.

Chapter 5

Future Research

I am convinced that *personal data should be controlled and stored by the individual who produces it*, rather than being practically monopolized by data brokers and tech giants. This view is in line with recent calls in favor of a more "decentralized Internet" (Berners-Lee, 2018) and various initiatives that promote the idea of self-sovereign data (see e.g., the Solid project and the Sovrin Foundation).[1] Regaining control of one's personal data is now being facilitated by the GDPR's right for citizens to retrieve a copy of the data collected about them by any platform, and by the ongoing development of secure personal data management systems (Anciaux et al., 2019). This context provides a timely opportunity to *develop machine learning frameworks that can learn from a huge number of decentralized personal datasets in a privacy-by-design manner*. In the long run, such approaches will help to *unlock the power of machine learning for applications that are fueled by large-scale and highly sensitive individual level data*, ranging from industrial applications such as home assistants and smart energy meters to research studies in fields like medicine, mobility, economics and the social sciences. They can also *make ML and AI accessible to citizens for collaborative computations at local, national and worldwide levels* by keeping individuals in control of their personal data and enabling them to decide how it is used and for which purpose. The work presented in this habilitation thesis, and more broadly the recent advances in decentralized and privacy-preserving ML, are steps towards the above objective. Nevertheless, many research, technological and regulatory challenges must be addressed before one could hope for broad adoption.

Below, I translate the above long-term vision into a short to mid-term research project focused on the design of scalable and privacy-preserving algorithms for learning from decentralized personal datasets, with the main objective of *narrowing the utility gap with the trusted curator setting*. This project will be funded in part by a 4-year grant from the French National Research Agency (ANR) that I was awarded in 2020.

[1] See https://solid.mit.edu/ and https://sovrin.org/

Overview. I will mostly consider fully decentralized algorithms, as they appear to be well-suited to collaborative computations over a large number of personal datasets: they scale nicely with the number of participants and avoid the costs of setting up and running a powerful central server. The general theme underlying the proposed research is to *strengthen differential privacy guarantees at no additional cost in utility by exploiting the way individual contributions are communicated within decentralized ML algorithms*. More precisely, I will consider the following three complementary research objectives:

1. broaden the scope of "privacy amplification by decentralization", proving that popular decentralized ML algorithms operating on general topologies naturally reinforce differential privacy guarantees;

2. propose novel algorithms at the intersection of decentralized ML and secure multiparty computation which scale to an arbitrary number of participants;

3. design data-adaptive communication schemes to speed up the convergence of decentralized ML algorithms operating on a large number of heterogeneous datasets.

Broadening the scope of privacy amplification by decentralization. I believe that the results presented in Section 4.2 of Chapter 4, which show that some decentralized algorithms achieve significantly better privacy-utility trade-offs than what is captured by local DP, are quite promising. I plan to push this line of research further and extend these initial results in several respects. A first objective is to consider general topologies beyond the ring and the complete graph to strike a good balance between privacy, scalability and robustness. Graph theoretic notions like the hitting time can provide adequate tools to quantify the relation between privacy gains and structural properties of the graph. Note that in an arbitrary graph, users are likely to leak more information to closer peers than to distant ones. Therefore, obtaining nontrivial privacy gains will require to consider appropriate relaxations of DP, such as metric-based DP (Andrés et al., 2013; Chatzikokolakis et al., 2013) or Pufferfish (Kifer and Machanavajjhala, 2014; Song et al., 2017). Another approach to mitigate such asymmetries is to allow the topology to evolve over time, which can also help to improve robustness to collusions for sparse topologies. A second objective is to prove privacy amplification guarantees for decentralized algorithms that allow more parallel computations across users (and are therefore more efficient). A natural extension of the algorithms we studied is to consider multiple tokens walking on the graph in parallel. I also plan to study randomized gossip protocols (Boyd et al., 2006), which are very efficient and quite popular for decentralized ML (see e.g., Lian et al., 2018; Hendrikx et al., 2019). Our preliminary work in the context of rumor spreading protocols **[AB-Conf4]** shows that gossip dissemination of a message provides differential privacy guarantees with respect to the identity of the sender. I plan to explore the possible connection with privacy amplification by shuffling (Erlingsson et

al., 2019), as gossip communication could be seen as providing approximate anonymity (or approximate shuffling when each user disseminates a message in parallel). Finally, I would like to study the fundamental limits of network DP in the different settings to establish the tightness of the analysis and hopefully identify decentralized protocols that provide optimal amplification.

In the ideas outlined above, one considers the conservative assumption that parties do not trust anyone. In many real-world scenarios however, *each participant may personally trust a small number of other peers in the network* (e.g., family members, close friends, referring physician...). Trust means that information sent by a party to a trusted peer does not need to satisfy DP (i.e., it is excluded from the view of the adversary), and could therefore be processed in the clear. Local trust naturally fits the fully decentralized framework, as the trust network (in which parties are connected if they trust each other) can be seen as an overlay over the communication network. Leveraging this trust network has the potential to further strengthen privacy amplification effects, but requires to deeply rethink the protocols and formal analyses. For instance, one should guarantee that the algorithms do not "propagate the trust" by transitivity to non-trusted parties (unlike what may happen in Facebook's image tagging system). I plan to study the effect of different trust models and topologies (e.g., pairwise symmetric or asymmetric, community-based) on the privacy guarantees.

Secure multi-party computation meets decentralized algorithms. Secure multi-party computation (MPC) can in principle simulate the trusted curator of the centralized setting but scales poorly with the number of parties, preventing its use for computations involving all n participants. Strikingly, though, decentralized communication protocols naturally break down the full problem into many steps involving a much smaller number of parties (as few as two in gossip protocols). This motivates the use of MPC at intermediate steps to reduce the noise needed for DP. I propose to investigate *how MPC may work in concert with decentralized ML to improve utility with additional complexity independent of the total number of participants*. Characterizing the trade-offs between utility and complexity to identify the sweet spots however requires to explore the huge combined search space of decentralized protocols, ML algorithms and MPC primitives.

I propose to study a crucial parameter: the *local scale at which MPC is used in decentralized ML algorithms*. At one extreme (local scale n) lies the high-utility but intractable simulation of the trusted curator involving all n participants, while lightweight but inaccurate DP-only decentralized ML approaches correspond to local scale 1. To explore intermediate points on this trade-off line, I propose to reduce the search space of MPC primitives by first focusing on secure aggregation with noise addition, which only reveals the differentially private aggregated output (see e.g., Jayaraman et al., 2018). With this primitive at hand, we can design variants of gossip SGD algorithms (Lian et al., 2018) in which the parameter aggregation between pairs of parties is done with MPC. However,

the utility gains are quite limited as MPC is only applied at local scale 2 (leading to a factor $\sqrt{2}$ reduction only in the standard deviation of the noise). Going beyond two parties is however challenging with existing decentralized ML algorithms. For instance, MPC must be run for each possible weighted average involved in (non-gossip) decentralized SGD (Lian et al., 2017), which ruins the benefits of MPC as each party learns more outputs than necessary. To go around this problem, I plan to build upon a flexible generalization of gossip protocols recently introduced by Loizou and Richtárik (2016) and Loizou and Richtárik (2019). Based on this framework, the goal is to design novel optimization algorithms that rely only on uniform averaging over a flexible number $2 \leq s \leq n$ of parties so as to achieve the desired trade-off between utility and complexity in time and communication. A precise analysis of convergence and complexity will be needed to quantify such trade-offs optimally.

I also aim to *extend the scope of the above approach to ML problems for which secure aggregation is too restrictive*, such as robust aggregation schemes for reducing the influence of byzantine updates by malicious parties (Blanchard et al., 2017). I plan to consider secure shuffling (Erlingsson et al., 2019) but also tailored MPC primitives that may provide better utility and/or lower computational cost for specific tasks.

Data-adaptive decentralized communication protocols. Data heterogeneity across users is known to slow down the convergence of decentralized learning protocols, which in turn damages the utility since the privacy loss composes across iterations. I propose to address these issues by *designing protocols where communication between parties adapts to the content of previous exchanges* so that users can adjust the weight they give to contributions sent by other peers. To the best of my knowledge, the design of optimal schemes for data-adaptive communication has been seldom studied: the network topology and associated mixing matrix W are typically chosen in a data-independent fashion.

I propose to start with a simplified setting in which the pairwise divergences between local data distributions are known in order to provide a proof of concept and quantify the expected gains. In recent preliminary work [**AB-Preprint1**], we have shown empirically that an appropriate data-driven topology can compensate for imbalanced class distributions across users. The goal here is to tackle more general kinds of heterogeneity in a principled manner by *using the divergences to design a communication weight matrix W which optimizes the convergence speed of the decentralized algorithm*. To accelerate consensus to a common model, the general intuition is that two parties with a large divergence should have more weight. I plan to formulate the problem of finding the best W as a constrained optimization problem (taking inspiration from Boyd et al., 2006, who only considered communication constraints), design decentralized algorithms to solve it, and prove that the resulting W leads to faster convergence. Several divergences between local distributions can be considered, from simple proxy measures such as the distance between local models trained in isolation or between a set of gradients (Karimireddy et al., 2020),

to complex divergences like Maximum Mean Discrepancy or the Wassertein distance. I will also study how the proposed methods can be combined with so-called "accelerated" gossip schemes (Cavalcante et al., 2011; Liu et al., 2013; Scaman et al., 2017).

In practical scenarios, divergences between local datasets are not known in advance and must be estimated from messages received from other parties. The goal here is to *adapt W dynamically along iterations while training the models*. Interestingly, there exists a rich literature on decentralized averaging and optimization with time-varying networks showing that convergence can be reached under very mild conditions on the sequence of matrices (see e.g., Nedic and Ozdaglar, 2009; Koloskova et al., 2020). This gives considerable freedom in the design of the updates schemes. To avoid long term dependencies, I plan to focus on order 1 Markov chains to make $W^{(t+1)}$ depend only on $W^{(t)}$ and the new information received at time t, which can be the observed discrepancy between updates communicated by the parties.

Finally, I will extend the above approaches to learn personalized models. While the approach presented in Chapter 2 provides a good starting point, I would like to investigate formulations based on clear statistical assumptions on the relation between local data distributions. For instance, the local distributions could be clustered or generated from a mixture of underlying distributions, as explored in a recent work **[AB-Conf1]**. This would open the way to the derivation of generalization bounds showing that when distributions are related enough, the empirical divergences give enough information to identify the underlying relations and therefore that the proposed approaches improve the generalization performance.

Integration, validation and implementation. The contributions of the above three scientific objectives are largely complementary. I believe that *integrating them into one system will yield the largest utility gains*. I will carefully analyze how to build solutions with optimal privacy-utility trade-off and low computational complexity, avoiding minor tensions that might exist between some of the results (e.g., adaptive weights may slightly reduce privacy amplification effects).

To support their strong theoretical guarantees, the proposed approaches will be *empirically validated to demonstrate that they provide significant practical improvements* in settings which approximate real-world use cases as realistically as possible. To avoid logistical and ethical issues, I plan to *simulate the decentralized scenario by creating many small personal datasets from public individual-level datasets*, e.g., a medical dataset including the characteristics of 100,000 patients[2] and a purchase log dataset which contains the history of 300,000 customers.[3] Existing decentralized learning benchmarks such as LEAF (Caldas et al., 2019) may also be used. The main objective will be to measure the trade-off between privacy and utility achieved by the proposed approaches compared to (i) a differentially

[2] https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008
[3] https://www.kaggle.com/c/acquire-valued-shoppers-challenge/

private model trained on the centralized data, and (ii) decentralized ML with baseline DP. We expect to largely outperform (ii) and to approach the utility of (i). Utility will be measured using prediction accuracy on unseen data. Privacy will be quantified by the parameters (ϵ, δ) of DP, and by the performance of membership inference attacks (Nasr et al., 2019) which provide a more concrete measure of protection. The impact of key problem dimensions (number of users, number of model parameters, size of the personal datasets, network topology) on the performance will also be evaluated.

The implementations and reproducible benchmarks will be distributed as open-source packages shared on public repositories, building upon my previous experience in developing high-quality packages. I also aim to disseminate the work in larger privacy-preserving ML open-source projects for increased visibility and impact. For instance, I intend to leverage existing contacts with the OpenMined community which develops the PySyft library.[4]

[4] https://github.com/OpenMined/PySyft

List of Scientific Contributions

Books

[AB-Book1] Bellet, A., Habrard, A., and Sebban, M. (2015a). *Metric Learning*. Morgan & Claypool Publishers (pp. 15, 35).

Articles in International Journals

[AB-Journal1] Bellet, A., Denis, P., Gilleron, R., Keller, M., and Vauquier, N. (2021a). "Pour plus de transparence dans l'analyse automatique des consultations ouvertes : leçons de la synthèse du Grand Débat National". *Statistique et Société*. To appear.

[AB-Journal2] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2021b). "Advances and Open Problems in Federated Learning". *Foundations and Trends® in Machine Learning* 14.1–2, pp. 1–210 (pp. 9–11, 15, 35, 37, 70).

[AB-Journal3] Paige, B., Bell, J., Bellet, A., Gascón, A., and Ezer, D. (2021). "Reconstructing Genotypes in Private Genomic Databases from Genetic Risk Scores". *Journal of Computational Biology* 28.5, pp. 435–451 (pp. 10, 15).

[AB-Journal4] de Vazelhes, W., Carey, C., Tang, Y., Vauquier, N., and Bellet, A. (2020). "metric-learn: Metric Learning Algorithms in Python". *Journal of Machine Learning Research* 21.138, pp. 1–6 (p. 15).

[AB-Journal5] Liu, K. and Bellet, A. (2019). "Escaping the Curse of Dimensionality in Similarity Learning: Efficient Frank-Wolfe Algorithm and Generalization Bounds". *Neurocomputing* 333, pp. 185–199.

[AB-Journal6] May, A., Garakani, A. B., Lu, Z., Guo, D., Liu, K., Bellet, A., Fan, L., Collins, M., Hsu, D., Kingsbury, B., Picheny, M., and Sha, F. (2019). "Kernel Approximation Methods for Speech Recognition". *Journal of Machine Learning Research* 20.59, pp. 1–36 (p. 15).

[AB-Journal7] Zheng, W., Bellet, A., and Gallinari, P. (2018). "A Distributed Frank-Wolfe Framework for Learning Low-Rank Matrices with the Trace Norm". *Machine Learning* 107.8–10, pp. 1457–1475 (p. 15).

[AB-Journal8] Bellet, A., Bernabeu, J. F., Habrard, A., and Sebban, M. (2016). "Learning Discriminative Tree Edit Similarities for Linear Classification — Application to Melody Recognition". *Neurocomputing* 214, pp. 155–161.

[AB-Journal9] Clémençon, S., Colin, I., and Bellet, A. (2016). "Scaling-up Empirical Risk Minimization: Optimization of Incomplete U-statistics". *Journal of Machine Learning Research* 17.76, pp. 1–36 (pp. 15, 35, 36, 40).

[AB-Journal10] Bellet, A. and Habrard, A. (2015). "Robustness and Generalization for Metric Learning". *Neurocomputing* 151.1, pp. 259–267.

[AB-Journal11] Bellet, A., Habrard, A., Morvant, E., and Sebban, M. (2014). "Learning A Priori Constrained Weighted Majority Votes". *Machine Learning* 97.1-2, pp. 129–154.

[AB-Journal12] Bellet, A., Habrard, A., and Sebban, M. (2012a). "Good edit similarity learning by loss minimization". *Machine Learning* 89.1, pp. 5–35.

[AB-Journal13] Bellet, A., Bernard, M., Murgue, T., and Sebban, M. (2010). "Learning state machine-based string edit kernels". *Pattern Recognition* 43, pp. 2330–2339.

Articles in Peer-Reviewed International Conferences

[AB-Conf1] Marfoq, O., Neglia, G., Bellet, A., Kameni, L., and Vidal, R. (2021). "Federated Multi-Task Learning under a Mixture of Distributions". *NeurIPS* (p. 83).

[AB-Conf2] Vogel, R., Bellet, A., and Clémençon, S. (2021). "Learning Fair Scoring Functions: Bipartite Ranking under ROC-based Fairness Constraints". *AISTATS* (pp. 15, 35, 41, 47).

[AB-Conf3] Bell, J., Bellet, A., Gascón, A., and Kulkarni, T. (2020a). "Private Protocols for U-Statistics in the Local Model and Beyond". *AISTATS* (pp. 14, 36, 37, 46, 50, 52, 55).

[AB-Conf4] Bellet, A., Guerraoui, R., and Hendrikx, H. (2020). "Who started this rumor? Quantifying the natural differential privacy guarantees of gossip protocols". *DISC* (pp. 15, 80).

[AB-Conf5] Maouche, M., Srivastava, B. M. L., Vauquier, N., Bellet, A., Tommasi, M., and Vincent, E. (2020). "A Comparative Study of Speech Anonymization Metrics". *INTERSPEECH* (p. 15).

[AB-Conf6] Paige, B., Bell, J., Bellet, A., Gascón, A., and Ezer, D. (2020). "Reconstructing Genotypes in Private Genomic Databases from Genetic Risk Scores". *RECOMB* (pp. 10, 15).

[AB-Conf7] Srivastava, B. M. L., Tomashenko, N., Wang, X., Vincent, E., Yamagishi, J., Maouche, M., Bellet, A., and Tommasi, M. (2020a). "Design Choices for X-vector Based Speaker Anonymization". *INTERSPEECH* (p. 15).

[AB-Conf8] Srivastava, B. M. L., Vauquier, N., Sahidullah, M., Bellet, A., Tommasi, M., and Vincent, E. (2020b). "Evaluating Voice Conversion-based Privacy Protection against Informed Attackers". *ICASSP* (p. 15).

[AB-Conf9] Zantedeschi, V., Bellet, A., and Tommasi, M. (2020). "Fully Decentralized Joint Learning of Personalized Models and Collaboration Graphs". *AISTATS* (pp. 13, 18, 25, 26, 28, 32).

[AB-Conf10] Srivastava, B. M. L., Bellet, A., Tommasi, M., and Vincent, E. (2019). "Privacy-Preserving Adversarial Representation Learning in ASR: Reality or Illusion?" *INTERSPEECH* (p. 15).

[AB-Conf11] Vogel, R., Bellet, A., Clémençon, S., Jelassi, O., and Papa, G. (2019). "Trade-offs in Large-Scale Distributed Tuplewise Estimation and Learning". *ECML/PKDD* (pp. 15, 37).

[AB-Conf12] Ailem, M., Zhang, B., Bellet, A., Denis, P., and Sha, F. (2018). "A Probabilistic Model for Joint Learning of Word Embeddings from Texts and Images". *EMNLP* (p. 15).

[AB-Conf13] Bellet, A., Guerraoui, R., Taziki, M., and Tommasi, M. (2018). "Personalized and Private Peer-to-Peer Machine Learning". *AISTATS* (pp. 13, 18, 24, 30–32).

[AB-Conf14] Vogel, R., Bellet, A., and Clémençon, S. (2018). "A Probabilistic Theory of Supervised Similarity Learning for Pointwise ROC Curve Optimization". *ICML* (p. 15).

[AB-Conf15] Vanhaesebrouck, P., Bellet, A., and Tommasi, M. (2017). "Decentralized Collaborative Learning of Personalized Models over Networks". *AISTATS* (pp. 13, 18, 24, 31, 32).

[AB-Conf16] Colin, I., Bellet, A., Salmon, J., and Clémençon, S. (2016). "Gossip Dual Averaging for Decentralized Optimization of Pairwise Functions". *ICML* (pp. 14, 36, 37, 43, 44, 52).

[AB-Conf17] Lu, Z., Guo, D., Bagheri Garakani, A., Liu, K., May, A., Bellet, A., Fan, L., Collins, M., Kingsbury, B., Picheny, M., and Sha, F. (2016). "A Comparison Between Deep Neural Nets and Kernel Acoustic Models for Speech Recognition". *ICASSP*.

[AB-Conf18] Papa, G., Bellet, A., and Clémençon, S. (2016). "On Graph Reconstruction via Empirical Risk Minimization: Fast Learning Rates and Scalability". *NIPS* (p. 35).

[AB-Conf19] Bellet, A., Liang, Y., Garakani, A. B., Balcan, M.-F., and Sha, F. (2015b). "A Distributed Frank-Wolfe Algorithm for Communication-Efficient Sparse Learning". *SDM* (p. 15).

[AB-Conf20] Clémençon, S., Bellet, A., Jelassi, O., and Papa, G. (2015). "Scalability of Stochastic Gradient Descent based on "Smart" Sampling Techniques". *INNS-BigData*.

[AB-Conf21] Colin, I., Bellet, A., Salmon, J., and Clémençon, S. (2015). "Extending Gossip Algorithms to Distributed Estimation of U-statistics". *NIPS* (pp. 14, 36, 39, 40, 52).

[AB-Conf22] Liu, K., Bellet, A., and Sha, F. (2015). "Similarity Learning for High-Dimensional Sparse Data". *AISTATS*.

[AB-Conf23] Papa, G., Clémençon, S., and Bellet, A. (2015). "SGD Algorithms based on Incomplete U-statistics: Large-Scale Minimization of Empirical Risk". *NIPS* (pp. 15, 36).

[AB-Conf24] Shi, Y., Bellet, A., and Sha, F. (2014). "Sparse Compositional Metric Learning". *AAAI*.

[AB-Conf25] Bellet, A., Habrard, A., and Sebban, M. (2012b). "Similarity Learning for Provably Accurate Sparse Linear Classification". *ICML*.

[AB-Conf26] Bellet, A., Habrard, A., and Sebban, M. (2011a). "An Experimental Study on Learning with Good Edit Similarity Functions". *ICTAI*.

[AB-Conf27] Bellet, A., Habrard, A., and Sebban, M. (2011b). "Learning Good Edit Similarities with Generalization Guarantees". *ECML/PKDD*.

Preprints & Technical Reports

[AB-Preprint1] Bellet, A., Kermarrec, A.-M., and Lavoie, E. (2021b). *D-Cliques: Compensating for Data Heterogeneity with Topology in Decentralized Federated Learning*. Tech. rep. arXiv:2104.07365 (pp. 17, 82).

[AB-Preprint2] Ladjel, R., Anciaux, N., Bellet, A., and Scerri, G. (2021). *Mitigating Leakage from Data Dependent Communications in Decentralized Computing using Differential Privacy*. Tech. rep. arXiv:2112.12411 (p. 15).

[AB-Preprint3] Mangold, P., Bellet, A., Salmon, J., and Tommasi, M. (2021). *Differentially Private Coordinate Descent for Composite Empirical Risk Minimization*. Tech. rep. arXiv:2110.11688 (p. 15).

[AB-Preprint4] Maouche, M., Srivastava, B. M. L., Vauquier, N., Bellet, A., Tommasi, M., and Vincent, E. (2021). *Enhancing Speech Privacy with Slicing*. Tech. rep. hal-03369137 (p. 15).

[AB-Preprint5] Noble, M., Bellet, A., and Dieuleveut, A. (2021). *Differentially Private Federated Learning on Heterogeneous Data*. Tech. rep. arXiv:2111.09278 (p. 18).

[AB-Preprint6] Cyffers, E. and Bellet, A. (2020). *Privacy Amplification by Decentralization*. Tech. rep. arXiv:2012.05326 (pp. 14, 58, 68–76).

[AB-Preprint7] Sabater, C., Bellet, A., and Ramon, J. (2020). *An Accurate, Scalable and Verifiable Protocol for Federated Differentially Private Averaging*. Tech. rep. arXiv:2006.07218 (pp. 14, 58, 63–67).

[AB-Preprint8] Dellenbach, P., Bellet, A., and Ramon, J. (Mar. 2018). *Hiding in the Crowd: A Massively Distributed Algorithm for Private Averaging with Malicious Adversaries*. Tech. rep. arXiv:1803.09984.

[AB-Preprint9] Lu, Z., May, A., Liu, K., Garakani, A. B., Guo, D., Bellet, A., Fan, L., Collins, M., Kingsbury, B., Picheny, M., and Sha, F. (Nov. 2014). *How to Scale Up Kernel Methods to Be As Good As Deep Neural Nets*. Tech. rep. arXiv:1411.4000.

[AB-Preprint10] Bellet, A., Habrard, A., and Sebban, M. (June 2013). *A Survey on Metric Learning for Feature Vectors and Structured Data*. Tech. rep. arXiv:1306.6709 (p. 15).

Ph.D. Thesis

[AB-PhD1] Bellet, A. (2012). "Supervised Metric Learning with Generalization Guarantees". PhD thesis. University of Saint-Etienne.

External References

Abowd, J. M. (2018). "The U.S. Census Bureau Adopts Differential Privacy". *KDD* (p. 12).

Acharya, J., Orlitsky, A., Suresh, A. T., and Tyagi, H. (2015). "The Complexity of Estimating Rényi Entropy". *SODA* (p. 36).

Ács, G. and Castelluccia, C. (2011). "I Have a DREAM! (DiffeRentially privatE smArt Metering)". *Information Hiding* (p. 59).

Anciaux, N., Bonnet, P., Bouganim, L., Nguyen, B., Pucheral, P., Popa, I. S., and Scerri, G. (2019). "Personal Data Management Systems: The security and functionality standpoint". *Information Systems* 80, pp. 13–35 (p. 79).

Andrés, M. E., Bordenabe, N. E., Chatzikokolakis, K., and Palamidessi, C. (2013). "Geo-indistinguishability: differential privacy for location-based systems". *CCS* (p. 80).

Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. (2013). "A public domain dataset for human activity recognition using smartphones". *ESANN* (p. 34).

Assran, M., Loizou, N., Ballas, N., and Rabbat, M. (2019). "Stochastic Gradient Push for Distributed Deep Learning". *ICML* (pp. 18, 37).

Ayache, G. and El Rouayheb, S. (2021). "Private Weighted Random Walk Stochastic Gradient Descent". *IEEE Journal on Selected Areas in Information Theory* 2 (1), pp. 452–463 (p. 69).

Aysal, T. C., Yildiz, M. E., Sarwate, A. D., and Scaglione, A. (2009). "Broadcast Gossip Algorithms for Consensus". *IEEE Transactions on Signal Processing* 57.7, pp. 2748–2761 (p. 21).

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2012). "Optimization with Sparsity-Inducing Penalties". *Foundations and Trends® in Machine Learning* 4.1, pp. 1–106 (p. 41).

Balle, B., Barthe, G., and Gaboardi, M. (2018). "Privacy amplification by subsampling: tight analyses via couplings and divergences". *NeurIPS* (pp. 59, 73).

Balle, B., Bell, J., Gascón, A., and Nissim, K. (2019). "The Privacy Blanket of the Shuffle Model". *CRYPTO* (pp. 59, 71, 72, 74).

Balle, B., Bell, J., Gascón, A., and Nissim, K. (2020). "Private Summation in the Multi-Message Shuffle Model". *CCS* (p. 59).

Bassily, R., Nissim, K., Stemmer, U., and Thakurta, A. G. (2017). "Practical Locally Private Heavy Hitters". *NIPS* (pp. 37, 48, 50).

Bassily, R. and Smith, A. (2015). "Local, private, efficient protocols for succinct histograms". *STOC* (p. 37).

Bassily, R., Smith, A. D., and Thakurta, A. (2014). "Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds". *FOCS* (pp. 58, 75, 77).

Bekkerman, R., Bilenko, M., and Langford, J. (2011). *Scaling Up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press (p. 10).

Bell, J. H., Bonawitz, K. A., Gascón, A., Lepoint, T., and Raykova, M. (2020b). "Secure Single-Server Aggregation with (Poly)Logarithmic Overheads". *CCS* (p. 59).

Berger, P., Buchacher, M., Hannak, G., and Matz, G. (2018). "Graph Learning Based on Total Variation Minimization". *ICASSP* (p. 26).

Berners-Lee, T. (2018). *One Small Step for the Web...* https://medium.com/@timberners_lee/one-small-step-for-the-web-87f92217d085 (p. 79).

Biau, G. and Bleakley, K. (2006). "Statistical Inference on Graphs". *Statistics & Decisions* 24, pp. 209–232 (p. 35).

Blanchard, P., Mhamdi, E. M. E., Guerraoui, R., and Stainer, J. (2017). "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent". *NIPS* (p. 82).

Blom, G. (1976). "Some properties of incomplete U-statistics". *Biometrika* 63.3, pp. 573–580 (p. 51).

Blum, M. (1983). "Coin flipping by telephone a protocol for solving impossible problems". *ACM SIGACT News* 15.1, pp. 23–27 (p. 66).

Boissier, M., Lyu, S., Ying, Y., and Zhou, D.-X. (2016). "Fast Convergence of Online Pairwise Learning Algorithms". *AISTATS* (p. 36).

Bollobás, B. (2001). *Random Graphs (2nd edition)*. Cambridge University Press (p. 64).

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). "Practical Secure Aggregation for Privacy-Preserving Machine Learning". *CCS* (pp. 59, 60, 62, 63, 65).

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2006). "Randomized gossip algorithms". *IEEE Transactions on Information Theory* 52.6, pp. 2508–2530 (pp. 20, 36–39, 80, 82).

Bradley, A. P. (1997). "The use of the area under the ROC curve in the evaluation of machine learning algorithms". *Pattern Recognition* 30.7, pp. 1145–1159 (pp. 36, 47).

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V., and Talwalkar, A. (2019). "LEAF: A Benchmark for Federated Settings". *Workshop on Federated Learning for Data Privacy and Confidentiality* (p. 83).

Cavalcante, R. L. G., Rogers, A., and Jennings, N. R. (2011). "Consensus acceleration in multiagent systems with the Chebyshev semi-iterative method". *AAMAS* (p. 83).

Champion, J., Shelat, A., and Ullman, J. (2019). "Securely Sampling Biased Coins with Applications to Differential Privacy". *CCS* (p. 52).

Chan, T.-H. H., Shi, E., and Song, D. (2012a). "Optimal Lower Bound for Differentially Private Multi-party Aggregation". *ESA* (p. 57).

Chan, T.-H. H., Shi, E., and Song, D. (2012b). "Privacy-Preserving Stream Aggregation with Fault Tolerance". *Financial Cryptography* (pp. 48, 59).

Chatzikokolakis, K., Andrés, M. E., Bordenabe, N. E., and Palamidessi, C. (2013). "Broadening the Scope of Differential Privacy Using Metrics". *PETS* (p. 80).

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. (2011). "Differentially Private Empirical Risk Minimization". *Journal of Machine Learning Research* 12, pp. 1069–1109 (p. 12).

Chaum, D. and Pedersen, T. P. (1993). "Wallet Databases with Observers". *CRYPTO* (p. 67).

Chen, W.-N., Kairouz, P., and Ozgur, A. (2020). "Breaking the Communication-Privacy-Accuracy Trilemma". *NeurIPS* (p. 57).

Cheng, H.-P., Yu, P., Hu, H., Zawad, S., Yan, F., Li, S., Li, H. H., and Chen, Y. (2019). "Towards Decentralized Deep Learning with Differential Privacy". *CLOUD* (p. 59).

Cheu, A., Smith, A. D., Ullman, J., Zeber, D., and Zhilyaev, M. (2019). "Distributed Differential Privacy via Shuffling". *EUROCRYPT* (p. 59).

Chung, F. (1997). *Spectral Graph Theory*. American Mathematical Society (pp. 40, 43).

Clarkson, K. L. (2010). "Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm". *ACM Transactions on Algorithms* 6.4, pp. 1–30 (p. 24).

Clémençon, S. (2014). "A statistical view of clustering performance through the theory of U-processes". *Journal of Multivariate Analysis* 124, pp. 42–56 (p. 35).

Clémençon, S., Lugosi, G., and Vayatis, N. (2008). "Ranking and empirical risk minimization of U-statistics". *The Annals of Statistics* 36.2, pp. 844–874 (pp. 35, 36, 40).

Colin, I. (2016). "Adapting machine learning methods to U-statistics". PhD thesis. Télécom ParisTech (p. 43).

Cormode, G., Kulkarni, T., and Srivastava, D. (2018). "Marginal Release Under Local Differential Privacy". *SIGMOD* (p. 37).

Deng, Y., Kamani, M. M., and Mahdavi, M. (2020). "Adaptive Personalized Federated Learning". arXiv: 2003.13461 (p. 18).

Dimakis, A. G., Kar, S., Moura, J. M. F., Rabbat, M. G., and Scaglione, A. (2010). "Gossip Algorithms for Distributed Signal Processing". *Proceedings of the IEEE* 98.11, pp. 1847–1864 (p. 37).

Ding, B., Kulkarni, J., and Yekhanin, S. (2017). "Collecting Telemetry Data Privately". *NIPS* (p. 12).

Dingledine, R., Mathewson, N., and Syverson, P. (2004). *Tor: The second-generation onion router*. Tech. rep. Naval Research Lab Washington DC (p. 59).

Dinh, C. T., Tran, N. H., and Nguyen, T. D. (2020). "Personalized Federated Learning with Moreau Envelopes". *NeurIPS* (p. 19).

Dong, X., Thanou, D., Frossard, P., and Vandergheynst, P. (2016). "Learning Laplacian matrix in smooth graph signal representations". *IEEE Transactions on Signal Processing* 64.23, pp. 6160–6173 (p. 26).

Duarte, M. F. and Hu, Y. H. (2004). "Vehicle classification in distributed sensor networks". *Journal of Parallel and Distributed Computing* 64.7, pp. 826–838 (p. 34).

Duchi, J. C., Agarwal, A., and Wainwright, M. J. (2012). "Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling". *IEEE Transactions on Automatic Control* 57.3, pp. 592–606 (pp. 37, 42).

Duchi, J. C., Jordan, M. I., and Wainwright, M. J. (2013). "Local Privacy and Statistical Minimax Rates". *FOCS* (pp. 13, 29, 44, 45, 57).

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). "Our Data, Ourselves: Privacy Via Distributed Noise Generation". *EUROCRYPT* (pp. 52, 59).

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). "Calibrating Noise to Sensitivity in Private Data Analysis". *TCC* (p. 12).

Dwork, C., Naor, M., Pitassi, T., Rothblum, G. N., and Yekhanin, S. (2010a). "Pan-Private Streaming Algorithms". *ICS* (p. 72).

Dwork, C. and Roth, A. (2014). "The Algorithmic Foundations of Differential Privacy". *Foundations and Trends® in Theoretical Computer Science* 9.3–4, pp. 211–407 (pp. 12, 62, 101).

Dwork, C., Rothblum, G. N., and Vadhan, S. (2010b). "Boosting and Differential Privacy". *FOCS* (p. 70).

Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., and Thakurta, A. (2019). "Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity". *SODA* (pp. 59, 71, 72, 74, 80, 82).

Erlingsson, Ú., Pihur, V., and Korolova, A. (2014). "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response". *CCS* (p. 12).

Evans, D., Kolesnikov, V., and Rosulek, M. (2018). "A Pragmatic Introduction to Secure Multi-Party Computation". *Foundations and Trends® in Privacy and Security* 2.2-3, pp. 70–246 (p. 52).

Evgeniou, T. and Pontil, M. (2004). "Regularized multi-task learning". *KDD* (p. 20).

Fallah, A., Mokhtari, A., and Ozdaglar, A. (2020). "Personalized federated learning: A meta-learning approach". *NeurIPS* (pp. 18, 57).

Fanti, G., Pihur, V., and Erlingsson, Ú. (2016). "Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries". *PoPETs* (p. 12).

Feldman, V., McMillan, A., and Talwar, K. (2020). "Hiding Among the Clones: A Simple and Nearly Optimal Analysis of Privacy Amplification by Shuffling". arXiv: 2012.12803 (pp. 59, 71, 72, 74).

Feldman, V., Mironov, I., Talwar, K., and Thakurta, A. (2018). "Privacy Amplification by Iteration". *FOCS* (pp. 59, 72, 75).

Fenner, T. I. and Frieze, A. M. (1982). "On the connectivity of random m-orientable graphs and digraphs". *Combinatorica* 2.4, pp. 347–359 (p. 64).

Franck, C. and Großschädl, J. (2017). "Efficient Implementation of Pedersen Commitments Using Twisted Edwards Curves". *Mobile, Secure, and Programmable Networking* (p. 66).

Frank, M. and Wolfe, P. (1956). "An algorithm for quadratic programming". *Naval Research Logistics (NRL)* 3, pp. 95–110 (p. 24).

Fujisaki, E. and Okamoto, T. (1997). "Statistical zero knowledge protocols to prove modular polynomial relations". *CRYPTO* (p. 67).

Ghazi, B., Kumar, R., Manurangsi, P., and Pagh, R. (2020). "Private Counting from Anonymous Messages: Near-Optimal Accuracy with Vanishing Communication Overhead". *ICML* (p. 59).

Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. (2020). "An Efficient Framework for Clustered Federated Learning". *NeurIPS* (p. 18).

Girgis, A., Data, D., Diggavi, S., Kairouz, P., and Theertha Suresh, A. (2021). "Shuffled Model of Differential Privacy in Federated Learning". *AISTATS* (p. 19).

Goldreich, O. (1998). "Secure multi-party computation". *Manuscript. Preliminary version* (p. 61).

Goldstein, H. (1991). "Multilevel modelling of survey data". *Journal of the Royal Statistical Society. Series D (The Statistician)* 40.2, pp. 235–244 (p. 34).

Hanzely, F., Hanzely, S., Horváth, S., and Richtarik, P. (2020). "Lower bounds and optimal algorithms for personalized federated learning". *NeurIPS* (pp. 19, 57).

Hanzely, F. and Richtárik, P. (2020). "Federated Learning of a Mixture of Global and Local Models". arXiv: 2002.05516 (p. 19).

Heinrich, S. (2017). "Flash Memory in the emerging age of autonomy". *Flash Memory Summit* (p. 9).

Hendrikx, H., Bach, F. R., and Massoulié, L. (2019). "An Accelerated Decentralized Stochastic Proximal Algorithm for Finite Sums". *NeurIPS* (pp. 18, 37, 80).

Herschtal, A. and Raskutti, B. (2004). "Optimising area under the ROC curve using gradient descent". *ICML* (p. 41).

Hoeffding, W. (1948). "A class of statistics with asymptotically normal distribution". *Annals of Mathematics and Statistics* 19, pp. 293–325 (pp. 37, 44, 46).

Hsieh, K., Phanishayee, A., Mutlu, O., and Gibbons, P. B. (2020). "The Non-IID Data Quagmire of Decentralized Machine Learning". *ICML* (pp. 17, 18).

Huai, M., Wang, D., Miao, C., Xu, J., and Zhang, A. (2020). "Pairwise Learning with Differential Privacy Guarantees". *AAAI* (p. 36).

Huang, Z., Mitra, S., and Vaidya, N. (2015). "Differentially Private Distributed Optimization". *ICDCN* (p. 59).

Jaggi, M. (2013). "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization". *ICML* (pp. 24, 26).

Jayaraman, B., Wang, L., Evans, D., and Gu, Q. (2018). "Distributed Learning without Distress: Privacy-Preserving Empirical Risk Minimization". *NeurIPS* (pp. 59, 81).

Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M., and Steen, M. van (2007). "Gossip-based peer sampling". *ACM Transactions on Computer Systems* 25.3 (pp. 18, 21, 27).

Jiang, Y., Konečný, J., Rush, K., and Kannan, S. (2019). "Improving Federated Learning Personalization via Model Agnostic Meta Learning". arXiv: 1909.12488 (p. 18).

Jiang, Z., Balu, A., Hegde, C., and Sarkar, S. (2017). "Collaborative Deep Learning in Fixed Topology Networks". *NIPS* (p. 18).

Johansson, B., Rabi, M., and Johansson, M. (2009). "A Randomized Incremental Subgradient Method for Distributed Optimization in Networked Systems". *SIAM Journal on Optimization* 20.3, pp. 1157–1170 (p. 69).

Kairouz, P., Liu, Z., and Steinke, T. (2021a). "The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation". *ICML* (p. 59).

Kairouz, P., Oh, S., and Viswanath, P. (2014). "Extremal mechanisms for local differential privacy". *NIPS* (pp. 45, 71).

Kairouz, P., Oh, S., and Viswanath, P. (2015). "The Composition Theorem for Differential Privacy". *ICML* (p. 102).

Kallus, N. and Zhou, A. (2019). "The Fairness of Risk Scores Beyond Classification: Bipartite Ranking and the xAUC Metric". *NeurIPS* (p. 47).

Kalofolias, V. (2016). "How to learn a graph from smooth signals". *AISTATS* (pp. 26, 28).

Kar, P., Sriperumbudur, B. K., Jain, P., and Karnick, H. (2013). "On the Generalization Ability of Online Learning Algorithms for Pairwise Loss Functions". *ICML* (pp. 35, 36, 40).

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. (2020). "SCAFFOLD: Stochastic controlled averaging for federated learning". *ICML* (pp. 10, 17, 18, 57, 82).

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. D. (2008). "What Can We Learn Privately?" *FOCS* (pp. 13, 29, 44, 57).

Katz, J. and Lindell, Y. (2014). *Introduction to Modern Cryptography, Second Edition*. CRC Press (p. 61).

Kempe, D., Dobra, A., and Gehrke, J. (2003). "Gossip-Based Computation of Aggregate Information". *FOCS* (pp. 36, 37).

Kermarrec, A., Leroy, V., and Thraves, C. (2011). "Converging Quickly to Independent Uniform Random Topologies". *PDP* (p. 27).

Khodak, M., Balcan, M.-F. F., and Talwalkar, A. S. (2019). "Adaptive gradient-based meta-learning methods". *NeurIPS* (p. 18).

Kifer, D. and Machanavajjhala, A. (2014). "Pufferfish: A framework for mathematical privacy definitions". *ACM Transactions on Database Systems* 39.1, p. 3 (p. 80).

Koloskova, A., Loizou, N., Boreiri, S., Jaggi, M., and Stich, S. U. (2020). "A unified theory of decentralized SGD with changing topology and local updates". *ICML* (pp. 18, 37, 72, 83).

Koloskova, A., Stich, S., and Jaggi, M. (2019). "Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication". *ICML* (pp. 18, 37).

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). "Federated Learning: Strategies for Improving Communication Efficiency". arXiv: 1610.05492 (p. 18).

Koriche, F. (2018). "Compiling Combinatorial Prediction Games". *ICML* (p. 28).

Krivelevich, M. (2010). "Embedding spanning trees in random graphs". *SIAM Journal on Discrete Mathematics* 24.4 (p. 64).

Kulis, B. (2013). "Metric Learning: A Survey". *Foundations and Trends® in Machine Learning* 5.4, pp. 287–364 (p. 35).

Kulkarni, T., Cormode, G., and Srivastava, D. (2019). "Answering Range Queries Under Local Differential Privacy". *SIGMOD* (pp. 37, 48, 50).

Lacoste-Julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. (2013). "Block-Coordinate Frank-Wolfe Optimization for Structural SVMs". *ICML* (p. 26).

Lee, A. (1990). *U-statistics: Theory and practice*. New York: Marcel Dekker, Inc. (pp. 36–38, 44, 46).

Li, C., Zhou, P., Xiong, L., Wang, Q., and Wang, T. (2018). "Differentially Private Distributed Online Learning". *IEEE Transactions on Knowledge and Data Engineering* (p. 59).

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020). "Federated Optimization in Heterogeneous Networks". *MLSys* (pp. 18, 57).

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. (2017). "Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent". *NIPS* (pp. 10, 12, 18, 36, 72, 82).

Lian, X., Zhang, W., Zhang, C., and Liu, J. (2018). "Asynchronous Decentralized Parallel Stochastic Gradient Descent". *ICML* (pp. 18, 37, 80, 81).

Lindell, Y. and Pinkas, B. (2009). "A Proof of Security of Yao's Protocol for Two-Party Computation". *Journal of Cryptology* 22.2, pp. 161–188 (p. 52).

Liu, J., Anderson, B. D., Cao, M., and Morse, A. S. (2013). "Analysis of accelerated gossip algorithms". *Automatica* 49.4, pp. 873–883 (p. 83).

Loizou, N. and Richtárik, P. (2016). "A new perspective on randomized gossip algorithms". *GlobalSIP* (p. 82).

Loizou, N. and Richtárik, P. (2019). "Revisiting Randomized Gossip Algorithms: General Framework, Convergence Rates and Novel Block and Accelerated Protocols". arXiv: 1905.08645 (p. 82).

Mann, H. B. and Whitney, D. R. (1947). "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". *Annals of Mathematical Statistics* 18.1, pp. 50–60 (pp. 36, 38).

Mansour, Y., Mohri, M., Ro, J., and Suresh, A. T. (2020). "Three Approaches for Personalization with Applications to Federated Learning". arXiv: 2002.10619 (p. 18).

Mao, X., Yuan, K., Hu, Y., Gu, Y., Sayed, A. H., and Yin, W. (2020). "Walkman: A Communication-Efficient Random-Walk Algorithm for Decentralized Optimization". *IEEE Transactions on Signal Processing* 68, pp. 2513–2528 (p. 69).

Marfoq, O., Xu, C., Neglia, G., and Vidal, R. (2020). "Throughput-Optimal Topology Design for Cross-Silo Federated Learning". *NeurIPS* (p. 72).

Maurer, A. (2006). "The Rademacher Complexity of Linear Transformation Classes". *COLT* (p. 20).

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. (2017). "Communication-efficient learning of deep networks from decentralized data". *AISTATS* (pp. 10, 11, 18, 57).

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. (2018). "Learning Differentially Private Recurrent Language Models". *ICLR* (pp. 13, 19).

Mironov, I. (2017). "Rényi Differential Privacy". *CSF* (pp. 75, 102).

Mironov, I., Pandey, O., Reingold, O., and Vadhan, S. P. (2009). "Computational Differential Privacy". *CRYPTO* (p. 59).

Mohri, M., Sivek, G., and Suresh, A. T. (2019). "Agnostic Federated Learning". *ICML* (p. 18).

Mosk-Aoyama, D. and Shah, D. (2008). "Fast Distributed Algorithms for Computing Separable Functions". *IEEE Transactions on Information Theory* 54.7, pp. 2997–3007 (p. 37).

Nasr, M., Shokri, R., and Houmansadr, A. (2019). "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning". *IEEE Symposium on Security and Privacy* (pp. 10, 84).

Nedic, A. (2011). "Asynchronous broadcast-based convex optimization over a network". *IEEE Transactions on Automatic Control* 56.6, pp. 1337–1351 (p. 21).

Nedic, A. and Ozdaglar, A. E. (2009). "Distributed Subgradient Methods for Multi-Agent Optimization". *IEEE Transactions on Automatic Control* 54.1, pp. 48–61 (pp. 37, 83).

Neglia, G., Xu, C., Towsley, D., and Calbi, G. (2020). "Decentralized gradient methods: does topology matter?" *AISTATS* (p. 72).

Nesterov, Y. (2009). "Primal-dual subgradient methods for convex problems". *Mathematical Programming* 120.1, pp. 261–283 (pp. 36, 41).

Parikh, N. and Boyd, S. (2013). "Proximal algorithms". *Foundations and Trends® in Machine Learning* 1.3, pp. 1–108 (p. 42).

Pedersen, T. P. (1991). "Non-interactive and information-theoretic secure verifiable secret sharing". *CRYPTO* (p. 66).

Pelckmans, K. and Suykens, J. (2009). "Gossip Algorithms for Computing U-Statistics". *IFAC Workshop on Estimation and Control of Networked Systems*, pp. 48–53 (p. 53).

Ram, S., Nedić, A., and Veeravalli, V. (2010). "Distributed Stochastic Subgradient Projection Algorithms for Convex Optimization". *Journal of Optimization Theory and Applications* 147.3, pp. 516–545 (p. 37).

Ram, S., Nedić, A., and Veeravalli, V. (2009). "Incremental stochastic subgradient algorithms for convex optimization". *SIAM Journal on Optimization* 20.2, pp. 691–717 (p. 69).

Razaviyayn, M., Hong, M., and Luo, Z.-Q. (2013). "A Unified Convergence Analysis of Block Successive Minimization Methods for Nonsmooth Optimization". *SIAM Journal on Optimization* 23.2, pp. 1126–1153 (p. 21).

Richtárik, P. and Takác, M. (2014). "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function". *Mathematical Programming* 144.1-2, pp. 1–38 (pp. 21, 28).

Sattler, F., Müller, K.-R., and Samek, W. (2020). "Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints". *IEEE Transactions on Neural Networks and Learning Systems* (p. 18).

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. (2017). "Optimal Algorithms for Smooth and Strongly Convex Distributed Optimization in Networks". *ICML* (p. 83).

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. (2019). "Optimal Convergence Rates for Convex Distributed Optimization in Networks". *Journal of Machine Learning Research* 20.159, pp. 1–31 (pp. 18, 37).

Schnorr, C. P. (1991). "Efficient signature generation by smart cards". *Journal of Cryptology* 4.3, pp. 161–174 (p. 67).

Shah, D. (2009). "Gossip Algorithms". *Foundations and Trends® in Networking* 3.1, pp. 1–125 (p. 37).

Shalev-Shwartz, S. (2012). "Online Learning and Online Convex Optimization". *Foundations and Trends® in Machine Learning* 4.2, pp. 107–194 (p. 104).

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press (p. 11).

Shamir, O. and Zhang, T. (2013). "Stochastic Gradient Descent for Non-smooth Optimization: Convergence Results and Optimal Averaging Schemes". *ICML* (p. 75).

Shen, C. and Li, H. (2010). "On the dual formulation of boosting algorithms". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.12, pp. 2216–2231 (p. 24).

Shi, E., Chan, T.-H. H., Rieffel, E. G., Chow, R., and Song, D. (2011). "Privacy-Preserving Aggregation of Time-Series Data". *NDSS* (pp. 59, 64).

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). "Membership Inference Attacks Against Machine Learning Models". *IEEE Symposium on Security and Privacy* (p. 10).

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. (2017). "Federated Multi-Task Learning". *NIPS* (p. 18).

Song, S., Chaudhuri, K., and Sarwate, A. D. (2013). "Stochastic gradient descent with differentially private updates". *GlobalSIP* (pp. 58, 77).

Song, S., Wang, Y., and Chaudhuri, K. (2017). "Pufferfish Privacy Mechanisms for Correlated Data". SIGMOD (p. 80).

Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. (2018). "D2: Decentralized Training over Decentralized Data". *ICML* (pp. 18, 36, 72).

Tseng, P. (2001). "Convergence of a block coordinate descent method for nondifferentiable minimization". *Journal of Optimization Theory and Applications* 109.3, pp. 475–494 (p. 21).

Tseng, P. and Yun, S. (2009). "Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization". *Journal of Optimization Theory and Applications* 140.3, pp. 140–513 (pp. 21, 28).

Tsianos, K., Lawlor, S., and Rabbat, M. (2015). "Push-Sum Distributed Dual Averaging for convex optimization". *CDC* (p. 37).

Wang, C., Wang, Y., E, W., and Schapire, R. (2015). "Functional Frank-Wolfe Boosting for General Loss Functions". arXiv: 1510.02558 (p. 24).

Wang, D., Gaboardi, M., and Xu, J. (2018). "Empirical Risk Minimization in Non-interactive Local Differential Privacy Revisited". *NeurIPS* (p. 57).

Wang, T., Blocki, J., Li, N., and Jha, S. (2017). "Locally Differentially Private Protocols for Frequency Estimation". *USENIX Security Symposium* (p. 37).

Watts, D. J. and Strogatz, S. H. (1998). "Collective dynamics of small-world networks". *Nature* 393.6684, pp. 440–442 (p. 52).

Wei, E. and Ozdaglar, A. (2012). "Distributed Alternating Direction Method of Multipliers". *CDC* (p. 37).

Wei, E. and Ozdaglar, A. E. (2013). "On the O(1/k) Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers". *GlobalSIP* (p. 24).

Wright, S. J. (2015). "Coordinate descent algorithms". *Mathematical Programming* 151.1, pp. 3–34 (pp. 21, 23, 28).

Xiao, L. (2010). "Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization". *Journal of Machine Learning Research* 11, pp. 2543–2596 (p. 41).

Xu, J., Zhang, W., and Wang, F. (2020). "A(DP)^{2}SGD: Asynchronous Decentralized Parallel Stochastic Gradient Descent with Differential Privacy". arXiv: 2008.09246 (p. 59).

Yağan, O. and Makowski, A. M. (2013). "On the Connectivity of Sensor Networks Under Random Pairwise Key Predistribution". *IEEE Transactions on Information Theory* 59.9, pp. 5754–5762 (p. 64).

Yang, Z., Lei, Y., Lyu, S., and Ying, Y. (2021). "Stability and Differential Privacy of Stochastic Gradient Descent for Pairwise Learning with Non-Smooth Loss". *ICML* (p. 36).

Yao, A. C.-C. (1986). "How to Generate and Exchange Secrets (Extended Abstract)". *FOCS* (p. 52).

Zhang, X., Khalili, M. M., and Liu, M. (2018). "Improving the Privacy and Accuracy of ADMM-Based Distributed Algorithms". *ICML* (p. 59).

Zhao, P., Hoi, S., Jin, R., and Yang, T. (2011). "Online AUC Maximization". *ICML* (p. 41).

Zheng, K., Mou, W., and Wang, L. (2017). "Collect at Once, Use Effectively: Making Non-interactive Locally Private Learning Possible". *ICML* (p. 57).

Appendix

A Reminders on Differential Privacy

In this appendix, we recall some standard properties and results pertaining to differential privacy that we use throughout the manuscript. These results, along with many others, can be found in the textbook by Dwork and Roth (2014).

To remain abstract, in the following we denote the data domain by $\mathcal{X}$ and a represent a dataset $\mathcal{D}$ as a member of $\mathbb{N}^{|\mathcal{X}|}$ (i.e., a histogram indicating the number of times each element of $\mathcal{X}$ occurs in $\mathcal{D}$).

A.1 Properties of Differential Privacy

Differential privacy comes with a number of properties that are instrumental in the design of complex private algorithms. We start with the postprocessing property, which states that a function evaluated on the output of a differentially private is itself differentially private (with the same parameters).

Theorem A.1 (Postprocessing)**.** *Let $\mathcal{A} : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{O}$ be (ϵ, δ)-DP and let $f : \mathcal{O} \to \mathcal{O}'$ be an arbitrary (randomized) function. Then $f \circ \mathcal{A} : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{O}'$ is (ϵ, δ)-DP.*

The composition property allows to control the cumulative privacy loss over multiple differentially private analyses of the same dataset. This holds even if the sequence of analyses is selected adaptively, i.e., if the next analysis is chosen based on the output of previous ones.

Theorem A.2 (Simple composition)**.** *Let $\mathcal{A}$ be an algorithm which takes as input a dataset $\mathcal{D}$ and outputs the result of a sequence $(\mathcal{A}_1(\mathcal{D}), \ldots, \mathcal{A}_K(\mathcal{D}))$ of K adaptively chosen algorithms. If at each step $k \in \{1, \ldots, K\}$, the selected algorithm $\mathcal{A}_k$ is guaranteed to satisfy (ϵ_k, δ_k)-DP, then $\mathcal{A}$ is (ϵ, δ)-DP with $\epsilon = \sum_{k=1}^{K} \epsilon_k$ and $\delta = \sum_{k=1}^{K} \delta_k$.*

Simple composition is known to be tight for pure ϵ-DP. For approximate (ϵ, δ)-DP with small enough ϵ, advanced composition gives a sublinear scaling of ϵ in K.

Theorem A.3 (Advanced composition). *Let $\mathcal{A}$ be an algorithm which takes as input a dataset $\mathcal{D}$ and outputs the result of a sequence $\mathcal{A}_1, \ldots, \mathcal{A}_K$ of K adaptively chosen algorithms on $\mathcal{D}$. If at each step $k \in \{1, \ldots, K\}$, the selected algorithm $\mathcal{A}_k$ is guaranteed to satisfy (ϵ, δ)-DP, then for any $\delta' > 0$ $\mathcal{A}$ is $(\epsilon', K\delta + \delta')$-DP with*

$$\epsilon' = \sqrt{2K \ln(1/\delta')}\epsilon + K\epsilon(e^{\epsilon} - 1).$$

The above advanced composition result was refined by Kairouz et al. (2015), allowing to compose over algorithms with different privacy budgets.

A.2 Laplace and Gaussian Mechanisms

A standard way to evaluate a function f in a differentially private fashion is via output perturbation, i.e., adding noise to the output of f. The simplest way to calibrate the noise to the global sensitivity of f. In the definition below, recall that $\sim$ denotes the neighboring relation used in the definition of DP.

Definition A.1 (Global ℓ_p sensitivity). *The global ℓ_p sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^K$ is given by*

$$\Delta_p(f) = \max_{\mathcal{D} \sim \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_p.$$

The Laplace mechanism adds centered Laplace noise whose scale is calibrated to the ℓ_1 sensitivity and the privacy parameter ϵ, guarantees pure ϵ-DP.

Theorem A.4 (Laplace mechanism). *Let $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^K$. Let $\eta = [\eta_1, \ldots, \eta_K] \in \mathbb{R}^K$ be a vector where each $\eta_k \sim \mathrm{Lap}(\Delta_1(f)/\epsilon)$ is drawn from the centered Laplace distribution with scale $\Delta_1(f)/\epsilon$. The algorithm $\mathcal{A}(\cdot) = f(\cdot) + \eta$ is ϵ-DP.*

The Gaussian mechanism uses Gaussian noise calibrated to the ℓ_2 sensitivity and the privacy parameters ϵ and δ. It guarantees (ϵ, δ)-DP.

Theorem A.5 (Gaussian mechanism). *Let $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^K$. Let $\eta = [\eta_1, \ldots, \eta_K] \in \mathbb{R}^K$ be a vector where each $\eta_k \sim \mathcal{N}(0, \sigma^2)$ is drawn from the centered Gaussian distribution with standard deviation $\sigma = \frac{\sqrt{2\ln(1.25/\delta)}\Delta_2(f)}{\epsilon}$. The algorithm $\mathcal{A}(\cdot) = f(\cdot) + \eta$ is (ϵ, δ)-DP.*

Note that for the specific case of the Gaussian mechanism, composition results tighter than those given in Theorem A.3 can be obtained through the framework of Rényi Differential Privacy (Mironov, 2017).

B Reminders on Notions of Regularity and Curvature

In this appendix, we recall some classic notions of regularity and curvature of functions that are routinely used in the design and analysis of optimization algorithms for machine learning (and their differentially private versions).

We assume throughout this section that the set $\mathcal{C}$ is a convex subset of $\mathbb{R}^p$. We first define Lipschitz continuity, which bounds how fast a function can change.

Definition B.1 (Lipschitz function)**.** *A function $f : \mathcal{C} \to \mathbb{R}$ is L-Lipschitz with respect to a norm $\|\cdot\|$ if for all $\theta, \theta' \in \mathcal{C}$, we have:*

$$|f(\theta) - f(\theta')| \leq L\|\theta - \theta'\|.$$

We can also define Lipschitzness with respect to blocks of coordinates, which is useful in the context of coordinate descent methods.

Definition B.2 (Block Lipschitz function)**.** *Let $p \in \mathbb{N}^*$ and $P_1, \ldots, P_b$ be a partition of the set $[\![p]\!]$ into b nonempty subsets of coordinates with $p_i = |P_i|$. Let $S_i \in p \times p_i$ be the column submatrix of the $p \times p$ identity matrix I_p corresponding to the coordinates in P_i, i.e., $(S_1, \ldots, S_b) = I_p$. The function $f : \mathbb{R}^p \to \mathbb{R}$ is called $(L_1, \ldots, L_b)$-block Lipschitz with respect to a norm $\|\cdot\|$ if for each $i \in [\![b]\!]$, we have for any $\theta, \tau \in \mathbb{R}^p$:*

$$|f(\theta + S_i\tau) - f(\theta)| \leq L_i\|\tau\|.$$

Smoothness refers to the fact that the gradient of a differentiable function is Lipschitz, which provides an upper bound for the function's curvature.

Definition B.3 (Smooth function)**.** *A differentiable function $f : \mathcal{C} \to \mathbb{R}$ is β-smooth with respect to a norm $\|\cdot\|$ if its gradient ∇f is β-Lipschitz, i.e., for all $\theta, \theta' \in \mathcal{C}$, we have:*

$$|\nabla f(\theta) - \nabla f(\theta')| \leq \beta\|\theta - \theta'\|.$$

If f is twice differentiable and β-smooth, then for any $\theta \in \mathcal{C}$, we have:

$$\nabla^2 f(\theta) \preceq \beta I.$$

The inverse of the smoothness constant gives the largest (constant) step size for which gradient descent converges. Block-wise smoothness constants can be defined from Definition B.2 and play a similar role in coordinate descent algorithms.

Convexity is a desirable property in optimization, as it ensures that any local minimum is also a global minimum.

Definition B.4 (Convex function)**.** *A function $f : \mathcal{C} \to \mathbb{R}$ is convex if for all $\theta, \theta' \in \mathcal{C}$ and $\alpha \in [0, 1]$, we have:*

$$f(\alpha\theta + (1-\alpha)\theta') \leq \alpha f(\theta) + (1-\alpha) f(\theta').$$

If f is differentiable, then f is convex if and only if for all $\theta, \theta' \in \mathcal{C}$ we have:

$$f(\theta') \geq f(\theta) + \langle \nabla f(\theta), \theta' - \theta \rangle.$$

Strong convexity further guarantees that the function has at most one local optimum (i.e., if it exists then it is a global optimum). It also provides a lower bound for a function's curvature.

Definition B.5 (Strongly convex function)**.** *A differentiable function $f : \mathcal{C} \to \mathbb{R}$ is μ-strongly convex with respect to some norm $\|\cdot\|$ if for all $\theta, \theta' \in \mathcal{C}$ we have:*

$$f(\theta') \geq f(\theta) + \langle \nabla f(\theta), \theta' - \theta \rangle + \frac{\mu}{2}\|\theta' - \theta\|^2.$$

If f is twice differentiable and μ-strongly convex, then for any $\theta \in \mathcal{C}$, we have:

$$\nabla^2 f(\theta) \succeq \mu I.$$

The condition number $\kappa = \beta/\mu$ of a β-smooth and μ-strongly convex function plays a key role in the convergence rate of first-order methods.

We conclude this section with a useful lemma showing an equivalence between (convex) Lipschitz functions and bounded gradients. This result is often used in differentially private optimization to bound the sensitivity of the gradients of Lipschitz loss functions.

Lemma B.1 (Shalev-Shwartz, 2012)**.** *Let $f : \mathcal{C} \to \mathbb{R}$ be a convex and differentiable function. Then f is L-Lipschitz with respect to norm $\|\cdot\|$ if and only if for all $\theta \in \mathcal{C}$ we have $\|\nabla f(\theta)\|_* \leq L$, where $\|\cdot\|_*$ is the dual norm.*